

El libro de L^AT_EX

El libro de L^AT_EX

Bernardo Cascales Salinas
Pascual Lucas Saorín
José Manuel Mira Ros
Antonio José Pallarés Ruiz
Salvador Sánchez-Pedreño Guillén

*Departamento de Matemáticas
Universidad de Murcia*



Datos de catalogación bibliográfica
CASCALES SALINAS, B., LUCAS SAORÍN, P., MIRA ROS, J. M., PALLARÉS RUIZ, A. J. y SÁNCHEZ-PEDREÑO GUILLÉN, S.
<i>El libro de LATEX</i>
PEARSON EDUCACIÓN, S.A., Madrid, 2003
ISBN 10: 84-205-3779-9
ISBN 13: 978-84-205-3779-5
Industria Gráfica 655
Formato: 195 · 250 mm
Páginas: 540

Todos los derechos reservados.

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (*arts. 270 y sgts. Código Penal*).

DERECHOS RESERVADOS

© 2003 por PEARSON EDUCACIÓN, S.A.
Ribera del Loira, 28
28042 Madrid (España)

CASCALES SALINAS, B., LUCAS SAORÍN, P., MIRA ROS, J. M., PALLARÉS RUIZ, A. J. y SÁNCHEZ-PEDREÑO GUILLÉN, S.

El libro de LATEX

ISBN 10: 84-205-3779-9

ISBN 13: 978-84-205-3779-5

Depósito Legal: M-17.856-2006

PEARSON PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN, S.A.

Última reimpresión, 2006

Equipo editorial:

Editora: Isabel Capella

Técnico editorial: Marta Caicoya

Equipo de producción:

Director: José Antonio Clares

Técnico: Isabel Muñoz

Diseño de cubierta: Juan Pedro Cascales Sandoval

Composición: COPIBOOK, S.L.

Impreso por: GRÁFICAS RÓGAR, S.A.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

Introducción

STE libro es un tratado acerca de \LaTeX . Si usted no está familiarizado con este término, se preguntará, sin duda, qué diablos es eso que tiene un curioso logotipo y un nombre que evoca cierto material plástico. Pues bien, \LaTeX es un conjunto de sentencias escritas en un lenguaje de programación llamado \TeX , nombre que se adopta también para denominar al intérprete de dicho lenguaje o compilador. Pero \TeX , el verdadero motor creado por Donald Ervin Knuth entre los años 1977 y 1978, no es un lenguaje de programación usual, sino uno orientado a la escritura de textos de excelente calidad. No se trata, naturalmente, de un editor de textos al estilo de otros bien conocidos y de cuyos nombres preferimos no acordarnos.

\TeX no es un editor de la familia WYSIWYG, término empleado para denominar a los editores que sólo trabajan sobre la pantalla del computador, dando un formato visual al texto, y en los cuales «lo que ves es lo que tienes». Muy al contrario, en \TeX usted escribe el texto acompañado de órdenes que el compilador, posteriormente, interpreta y ejecuta para proporcionar un texto perfectamente compuesto. Un solo ejemplo bastará: en los editores WYSIWYG, si desea escribir el carácter ∇ tendrá que buscarlo en un menú que se puede denominar «Insertar|Símbolos» o algo similar; en \TeX debe escribir $\$\\nabla\$$.

ESLIE LAMPORT creó \LaTeX en 1982, con la intención de simplificar la tarea de aquéllos que desean utilizar \TeX . El prólogo de *\LaTeX —A Document Preparation System*, [37], la obra en la que Lampert presenta su producto, comienza así¹:

\LaTeX (sistema para la preparación de documentos) es una versión especial de \TeX , programa creado por Donald Knuth. \TeX es un programa sofisticado diseñado para producir documentos de alta calidad, especialmente textos matemáticos. \LaTeX añade a \TeX una colección de comandos que simplifican el mecanografiado, permitiendo al usuario concentrarse en la estructura del texto, en vez de en los comandos para dar formato. Al transformar \TeX en \LaTeX , he tratado de convertir un coche de carreras muy bien afinado en un confortable sedán familiar. El sedán familiar no está pensado para ir tan rápido o ser tan excitante de conducir como el coche de carreras, pero es confortable y le llevará a la tienda de ultramarinos sin alborotos. Sin embargo, \LaTeX tiene toda la potencia de \TeX escondida bajo el capó, y el conductor más aventurero puede hacer con él todo lo que puede hacer con \TeX .

¹La traducción es de los autores de este libro

Posteriormente, \LaTeX ha ido actualizándose gracias al trabajo de un grupo de personas, reunidas en el denominado « \LaTeX Team». Hoy en día \LaTeX , o más precisamente $\text{\LaTeX} 2_{\epsilon}$, es un programa de unas 8 000 líneas de código \TeX , junto a un buen número de otros archivos con contenido relacionado.

J I ha llegado hasta aquí es seguro que usted está interesado en la composición de textos con ayuda del computador. Le podemos asegurar que \LaTeX es la herramienta ideal para esta tarea y nuestra intención, con este libro, es convencerle de ello, sin necesidad de que su esfuerzo sea desmesurado. Si se decide a conocer \LaTeX un poco más de cerca a través del libro que tiene en sus manos, deseamos que lo haga, desde el primer momento, con la práctica, sentado frente a su computador; si es posible, en un ambiente acogedor. Así es como deseamos que usted entre en este apasionante mundo.

Q UIZÁS se pregunte si quinientas páginas no son demasiadas para enseñar a manipular un «editor de textos», más aún conociendo, como conoce, otros programas que comparten este objetivo. El breve ejemplo anterior del carácter ∇ quizás no sea suficiente para hacerle captar la diferencia entre esos otros editores y el *compilador de textos* que nos brinda el tandem \TeX - \LaTeX . Algunas observaciones adicionales podrán darle una idea de que esto es otra cosa... \TeX ha sido considerado, por expertos en tipografía y edición, la mayor aportación a esta disciplina técnico-artística desde Gutenberg. Nuestra experiencia es que cada vez que hemos compuesto un documento y lo hemos enviado a un impresor profesional, éste ha quedado satisfecho del producto entregado y apenas ha tenido que trabajar para realizar pequeñas modificaciones en la composición. En varios países se han formado empresas de edición sobre la base de \TeX y \LaTeX . Finalmente, por favor, ojee este libro; si es usted conocedor de algunas normas tipográficas y de edición podrá emitir un juicio, provisional, sobre la composición general del mismo. Si su impresión es buena, y esperamos que así sea, debe saber que ha sido compuesto ¡completamente! por nosotros, sus autores, con herramientas de la familia \TeX - \LaTeX . ¿Son razones suficientes? Perdón, olvidábamos una: \TeX y \LaTeX , así como un sinfín de recursos complementarios, son gratuitos.

H ACE ya bastantes años que impartimos cursos sobre \LaTeX . Muchos de nuestros alumnos, en total más de quinientos, tienen la impresión inicial de que utilizar \LaTeX es muy difícil; posiblemente usted ya puede tener la misma idea, después del ya reiterativo ejemplo de $\$\backslash\nabla\$$. Nuestra experiencia es, sin embargo, totalmente opuesta. Creemos que \LaTeX , al menos a nivel de usuario común, no es especialmente difícil; tan sólo requiere paciencia y experiencia. No es una herramienta para el que va a utilizar el computador muy esporádicamente para escribir documentos, sino para el que lo utiliza con frecuencia. En su inicio, el uso de \TeX se circunscribía al ámbito científico, incluso universitario. Uno de nuestros objetivos, en todos estos años, ha sido difundir el uso de \LaTeX y facilitar el acceso al mismo; una dedicación que, en ocasiones, parece rozar el «apostolado». Actualmente estamos convencidos de que \LaTeX es una herramienta de enorme utilidad, independientemente del tipo de documentos que se escriban, a excepción, quizás, de periódicos y revistas de amplia y rápida difusión. La antítesis de un documento científico podría ser una novela; un novelista necesita muy pocos conocimientos de \LaTeX , tan pocos que puede aprenderlos en un día o dos; a cambio, los documentos que escriba en la elaboración de su novela serán de gran calidad, tipográfica se entiende, fácilmente adaptables a posibles cambios, transportables en soportes de muy poca capacidad y rápidamente transmisibles por correo electrónico.

○ N febrero del año 2000 aparecía publicado el libro *LATEX, una imprenta en sus manos*, escrito por nosotros mismos y publicado por AULA DOCUMENTAL DE INVESTIGACIÓN. Creemos que fue bien acogido; hemos recibido opiniones favorables de personas que lo han manejado con asiduidad, y se encuentra prácticamente agotado. Quizás este pequeño éxito provocó el interés de PEARSON EDUCACIÓN que, generosamente, nos ha ofrecido la posibilidad de publicar este «segundo libro de LATEX».

○ OS que conocen nuestro primer libro descubrirán que, en éste, la filosofía general ha cambiado enormemente. No posee un carácter enciclopédico, aproximándose más a un «libro de texto». La primera parte ofrece un proceso de aprendizaje, bastante amplio, en dieciocho lecciones eminentemente prácticas. Cada lección propone un buen número de ejercicios cuidadosamente seleccionados, algunos de ellos con soluciones. La segunda parte tiene una intención más próxima a un «manual de uso». En ella se profundiza en todos los aspectos tratados en las lecciones y en otros nuevos. Naturalmente hemos tratado de actualizar la información, introduciendo las mejores herramientas que conocemos entre las aparecidas en los últimos tres años.

○ L libro se ve enormemente enriquecido con la inclusión de un CD-ROM que contiene todos los programas y paquetes que hemos utilizado para escribirlo, y muchos más. La gratuitud de todas estas herramientas es, indudablemente, un valor añadido al empleo de LATEX, más aún en estos tiempos en los que cada día son menos y mayores empresas del ramo las que dominan el mercado. Se incluyen también en el disco las soluciones a algunos ejercicios y algunos documentos complementarios que no han tenido cabida en el texto impreso. El símbolo indica que en este disco se encuentra algún material relacionado con el texto en el que aparece. El CD-ROM posee un programa autoejecutable que le ayudará a acceder de forma cómoda al material que contiene.

○ ESEAMOS agradecer a PEARSON EDUCACIÓN el apoyo e interés que siempre nos han brindado. Es justo destacar a Jorge Martín, porque con él tuvimos las primeras conversaciones, entusiastas, de las que surgió la idea de llevar a cabo este proyecto; a Isabel Capella que, desde su puesto de responsabilidad, ratificó, con el mismo entusiasmo que Jorge, la propuesta de su compañero, y concretó los medios y plazos para llevarlo a término; y a Marta Caicoya, que siempre nos ha atendido con gran amabilidad y ha elogiado las versiones preliminares de este libro.

No podemos olvidar agradecer la colaboración de Juan Pedro Cascales Sandoval, autor de la portada y de los gráficos que encabezan las lecciones (salvo el de la lección 8), así como a nuestros alumnos de los distintos cursos sobre LATEX y al Departamento de Matemáticas de la Universidad de Murcia, que no ha dejado de proporcionarnos los medios necesarios para la elaboración de este texto.

○ AN sólo esperamos que usted, lector o lectora, disfrute tanto como nosotros utilizando LATEX, creando documentos de elevada calidad estética y resolviendo los problemas que, sin duda, terminarán planteándosele. Ojalá que nuestro propio entusiasmo al escribir estas páginas y el de nuestros nuevos editores al publicarlas, se vea acompañado por el interés de nuestros antiguos y nuevos lectores.

Convenciones utilizadas

*H*emos recurrido a algunas convenciones en lo que se refiere a los tipos utilizados, a fin de facilitar la lectura, permitiendo al lector distinguir, mediante el efecto diacrítico aportado por los cambios de tipo, elementos de carácter diferente, algo importante cuando se describen lenguajes formales. Estas convenciones se pueden resumir en las siguientes:

- Los tipos de «máquina de escribir» se utilizan para todo lo que debe ser escrito sin cambio alguno por el usuario, por ejemplo, comandos, nombres de ficheros, extensiones de éstos, etc.
- Los argumentos, es decir, textos variables que corresponde al usuario especificar, se indican en *italica*. En ocasiones se combina este efecto con el anterior, dando como resultado tipos de *máquina de escribir itálicos*, que representan textos que son parte del código pero que deben ser especificados por el usuario; un ejemplo típico es el del nombre de un comando definido por el usuario.
- Los tipos «sin adornos» se utilizan para los nombres de paquetes y opciones, tanto de las clases de documento como de los propios paquetes.
- Finalmente, las VERSALITAS se reservan para los nombres de programas (excepto algunos que poseen su propio logotipo) y para la mayor parte de formatos de archivos electrónicos.

Los ejemplos son una parte esencial del contenido; todos ellos muestran el código *LATEX*, es decir, lo que el usuario debe escribir, en la columna de la izquierda, y el resultado que éste produce en la columna de la derecha. Es preciso advertir que las medidas de longitud en esta parte derecha no se corresponden exactamente con las indicadas por el código, debido a que se ha recurrido a un proceso global de reducción de tamaño, motivado por cuestiones técnicas que no es necesario detallar aquí.

Sobre la composición de este libro

*H*emos compuesto este libro utilizando fuentes Times proporcionadas por el paquete *mathptmx* (véase la página 185). Si usted realiza los ejercicios de este libro, o simplemente compone un documento, con el preámbulo estándar que aparece en la figura I.2.5, se utilizarán fuentes Computer Modern en lugar de las fuentes Times.

El estilo con el que se ha maquetado este libro ha sido específicamente diseñado, para la ocasión, por los autores, creando para ello un fichero propio de *LATEX* que se ocupa de todos los pormenores que usted puede apreciar: antetítulos, cabeceras, ejemplos, sombreado, múltiples índices, lista de bibliografía, etc.

*P*ara resolver las muchas cuestiones gramaticales y ortográficas que nos han surgido al escribir, hemos recurrido con frecuencia al *Diccionario de la Lengua de la Real Academia Española*, y algunas veces hemos realizado consultas a la propia Real Academia. Para aconsejarnos y resolver nuestras muchas dudas tipográficas, y de ortografía técnica, hemos utilizado como referencia los libros de J. Martínez de Sousa [60, 61].

\\ O siendo, los autores, hombres de letras sino de números, acabaremos esta introducción con unas cuantas cifras que hablan, por sí mismas, del proceso de elaboración de este libro: hemos escrito unos 250 ficheros de entrada (código L^AT_EX), con un total aproximado de 3 MBytes; el fichero final de salida, en formato PostScript, incluyendo todos los gráficos y ejemplos especiales, presto para la impresión del libro, es de 22,3 MBytes; los dos índices terminológicos juntos suman más de 4 200 entradas, en tres niveles de profundidad; cada vez que se ordenan las entradas se realizan por encima de 52 000 comparaciones, eso sí, ¡en un par de segundos! ¡Qué cantidad de trabajo ha hecho L^AT_EX por nosotros! ¿no? ¿Se imagina tener que escribir este libro con uno de los editores cuyos nombres quisimos olvidar al principio de la introducción?

De todo lo anterior se desprende que somos nosotros, los autores, los únicos responsables de los errores, o posibles aciertos, en el contenido y la presentación de este libro.

Murcia, a 27 de agosto de 2003

Los autores

 Perdón ¡Casi se nos olvida! Todos los ficheros fuente comprimidos ocupan 834 Kbytes: un tamaño ridículo, incluso para viajar (velozmente ......) por Internet, tratándose de un libro de más de 500 páginas. ¡Ahora ya puede leer el libro, tranquilamente!

Índice general

Índice de figuras	XVII
Índice de cuadros	XVIII
I L^AT_EX básico en dieciocho lecciones	1
1. Cómo funciona L^AT_EX	3
Esquema básico de funcionamiento, 3. — Texto fuente, 3. — Composición o compilación, 4. — Visionado o impresión, 5. — Esquema real de funcionamiento, 5. — Documentos en formato PostScript, 6. — Documentos en formato PDF, 7. — Creando documentos PDF, 8.	
2. Primera cita: composición de un documento	10
Instalación de programas, 10. — ¿Qué contiene un documento fuente para L ^A T _E X?, 11. — Componiendo el primer documento, 13. — Preámbulo y cuerpo del manuscrito, 16. — El preámbulo para nuestras lecciones, 18. — Depurando errores, 18.	
3. Párrafos: alineación y párrafos especiales	22
Párrafos, sangría y saltos de línea, 22. — División silábica, 24. — Párrafos centrados o alineados por un solo lado, 25. — Párrafos especiales: citas textuales y poemas, 27. — Interlínea, 29.	
4. Caracteres reservados y signos ortográficos	31
Los diez caracteres reservados, 31. — Comentarios en el código fuente, 32. — Evitar la separación de palabras, 33. — Signos ortográficos, 34. — Comillas, 35. — Guiones, 36. — Puntos suspensivos, 36. — Ordinales y grados, 37. — Otros signos, 38. — Euros, 39.	
5. Tipos y colores	41
Diferentes familias de tipos: roman, sanserif y typewriter, 41. — Los perfiles de una familia: recto, itálico, inclinado y versalita, 42. — Grosor de los tipos: normal y negrita, 43. — Diferentes tamaños de letras, 44. — Los colores predefinidos, 46. — Escribiendo texto en colores, 46.	

6. Libros y artículos	48
Las clases de documento, 49. — Las unidades de estructura fundamentales, 50. — Portada, prólogos y apéndices, 52. — Grandes unidades de estructura en la clase book, 56. — Crear el índice general, 57. — Opciones básicas de la clase de documento, 58.	
7. Paginar un documento	63
Estilos de página, 64. — Parámetros de una página: anchura, altura y márgenes, 64. — Espacios verticales y horizontales, 67. — Saltos de página, 69.	
8. Referencias cruzadas	71
Las etiquetas de L ^A T _E X, 71. — Precauciones con el uso de las etiquetas, 73. — Creando enlaces de hipertexto con L ^A T _E X, 74. — El paquete hyperref, 74. — Algunas opciones del paquete hyperref, 74. — Cómo crear enlaces de hipertexto, 75.	
9. Inclusión de gráficos	78
Distintos tipos de gráficos, 79. — El paquete graphicx y los formatos gráficos incorporables, 80. — El comando para la inclusión de gráficos, 82. — Las figuras como objetos flotantes, 84.	
10. Órdenes y declaraciones: comandos y entornos	89
Grupos, 89. — Comandos, 90. — Entornos, 92. — Definiendo nuevos comandos y entornos, 93.	
11. Listas	97
Listas numeradas y listas con viñetas, 97. — Listas descriptivas, 98.	
12. Columnas	101
Escríptura en dos columnas, 101. — Varias columnas con el paquete multicol, 102. — Textos paralelos en columnas con multicols, 103.	
13. Notas	105
Notas a pie de página, 105. — Situaciones especiales, 107. — Marcas especiales para las notas a pie, 108. — Notas al margen, 108.	
14. Tablas	110
Tablas básicas, 110. — Algunos ejemplos, detalles y trucos, 114. — Textos que se extienden a varias columnas, 116. — Un nuevo objeto flotante: los cuadros, 117.	
15. Citas bibliográficas	121
Primeros ejemplos de listas bibliográficas, 121. — Diferentes formas de producir referencias bibliográficas, 122. — El entorno thebibliography, 123. — Citas y enlaces de hipertexto, 125. — Opciones de hyperref relativas a la bibliografía, 125.	

16. Fórmulas: una introducción	126
Matemáticas con el paquete <code>amsmath</code> , 126. — El modo matemático. Fórmulas, 127. — Superíndices y subíndices, 130. — Raíces, 131. — Fracciones y números combinatorios, 131. — Textos y espacios en fórmulas, 133. — Letras griegas, 134. — <code>LATEX</code> para químicos: el paquete <code>chemsym</code> , 135.	
17. Manejo de contadores y longitudes	139
Contadores: formatos, representación y gestión de valores, 139. — Longitudes y gestión de valores, 142.	
18. Cajas y marcos	146
Cajas y marcos, 146. — Pequeñas páginas en medio del texto, 148. — Pintar rayas, 150. — Reutilizar cajas, 151. — Distribuir el espacio entre cajas, 153.	
II Para ser un LATEXperto	157
1. Fundamentos	159
1.1. Detalles sobre espacios, líneas y páginas Espacios y signos de puntuación, 159. — Influyendo en la formación de los párrafos, 161. — Más ayuda en la paginación, 162.	159
1.2. Casi todo sobre <code>babel</code> Comandos básicos de <code>babel</code> , 166. — La traducción de antetítulos, 168. — Comandos para taquígrafos, 169.	164
1.3. Las lenguas del Estado español Métodos taquigráficos y algunos comandos extras, 171. — Comandos y acciones de la opción <code>spanish</code> , 173.	170
1.4. Gestión de tipos mediante paquetes Creación de tipos: Computer Modern Fonts y METAFONT, 177. — Fuentes EC y el paquete <code>fontenc</code> , 179. — Tipos PostScript, 180. — Las 35 fuentes PostScript de Adobe, 182. — Tabla de una fuente y acceso a los caracteres, 187. — Un tipo para cada ocasión, 192.	177
1.5. Más sobre contadores y longitudes Contadores, 195. — Longitudes, 196.	194
2. Estructura	203
2.1. La numeración de las unidades de estructura	203
2.2. Más sobre índices: general, de cuadros y de figuras Inclusión manual de entradas, 206. — Comandos frágiles y robustos en argumentos móviles, 208.	205
2.3. Compilación por trozos	209
2.4. El estilo de las páginas Las «marcas» de la cabecera y el pie, 212. — Estilos mejorados: el paquete <code>fancyhdr</code> , 217.	211

2.5. Parámetros de composición de páginas	224
2.6. Más opciones para las clases de documento	227
Imprimir en apaisado: la opción <code>landscape</code> , 227. — Algo más sobre la opción <code>draft</code> y los mensajes <code>Overfull</code> , 229.	
2.7. Índices terminológicos	231
El proceso manual en breve, 232. — El proceso automático con <code>MAKEINDEX</code> , 233. — Variaciones en la selección de entradas para el índice, 235. — <code>MAKEINDEX</code> , 240. — Estilos para <code>MAKEINDEX</code> , 241. — Pequeñas modificaciones en el entorno <code>theindex</code> , 244. — Varios índices terminológicos, 246. — Glosarios con <code>MAKEINDEX</code> , 248.	
2.8. Mecanizar la bibliografía	250
<code>BIBTEX</code> , 251. — Bases de datos para <code>BIBTEX</code> , 256. — Algunos paquetes útiles para el manejo de bibliografía, 266. — Varias listas de bibliografía, 268. — Glosarios con <code>BIBTEX</code> , 270.	
3. Construcciones	273
3.1. Nuevas posibilidades con el paquete <code>color</code>	273
Modelos y definición de colores, 274. — Sintaxis extendida para texto en color, 275. — Cajas y páginas en color, 275.	
3.2. Personalización de listas	276
Contadores, viñetas y etiquetas, 276. — El entorno <code>list</code> , 281.	
3.3. Control y extensiones de las tablas	288
Parámetros en las tablas, 288. — Ampliando las opciones del entorno <code>tabular</code> : el paquete <code>array</code> , 289. — Tablas grandes: el paquete <code>longtable</code> , 291. — Agrupar filas: el paquete <code>multirow</code> , 295. — Tablas con filetes dobles: el paquete <code>hhline</code> , 296. — Tablas con color: el paquete <code>colortbl</code> , 298.	
3.4. Sobre los objetos flotantes	301
Control de la ubicación, 301. — Incluir objetos al comienzo de una página: el paquete <code>afterpag</code> , 304. — Muchos formatos de leyendas con el paquete <code>caption2</code> , 305. — Pequeñas figuras rodeadas de texto, 309.	
3.5. Algo más sobre cajas	314
El manejo de las cajas en <code>TEX</code> , 317. — Más posibilidades para enmarcar y sobreescibir cajas, 326.	
4. Gráficos	331
4.1. Esbozo de una estrategia para la inclusión de gráficos	332
Algunos métodos de conversión de formatos gráficos, 333.	
4.2. El paquete <code>graphicx</code>	335
Opciones de <code>graphicx</code> , 335. — Todos los parámetros para la inclusión de gráficos externos, 336. — Rotación y escalado de objetos, 338.	
4.3. Más herramientas para rotar: el paquete <code>rotating</code>	340

4.4. Introducción a <i>PSTricks</i>	340
Nociones básicas, 341. — Algunos objetos gráficos, 342. — Parámetros de <i>PSTricks</i> , 353.	
— Paquetes de <i>PSTricks</i> , 355.	
5. Matemáticas	357
5.1. Recordando conceptos básicos	358
5.2. Opciones del paquete <i>amsmath</i>	358
5.3. Símbolos matemáticos	359
Negación de símbolos, 360. — Redefiniendo símbolos, 361.	
5.4. Fuentes en modo matemático	362
5.5. Miscelánea matemática	368
Acentos en modo matemático, 368. — Puntos suspensivos en matemáticas, 369. — Guienes que no parten expresiones, 370. — Tamaños de las fórmulas, 370. — Fórmulas encerradas, 371. — Espacios en fórmulas, 372. — Unos símbolos sobre otros, 372. — El comando <code>\smash</code> , 374. — Mejoras en las raíces, 374. — Más fracciones, 375. — Delimitadores: paréntesis, corchetes y llaves, 376.	
5.6. Matrices y determinantes	378
5.7. Fórmulas en varias líneas: alineamiento de ecuaciones	380
El entorno <code>eqnarray</code> , 380. — Comparación de entornos para ecuaciones múltiples, 382.	
— Fórmulas demasiado largas: los entornos <code>split</code> y <code>multline</code> , 383. — Varias ecuaciones en varias líneas: <code>gather</code> , <code>align</code> , <code>flalign</code> y <code>alignat</code> , 385. — Bloques de fórmulas, 386. — El entorno <code>cases</code> , 387. — Incluyendo texto en grupos de ecuaciones alineadas, 387. — Saltos de página en fórmulas de varias líneas, 388. — Numerando ecuaciones, 388.	
5.8. Nombres de funciones	389
5.9. Matemáticas y <i>babel</i> con opción <i>spanish</i>	391
5.10. Operadores de tamaño variable: integrales y sumatorios	392
Ubicación de los límites en operadores de tamaño variable, 393. — Subíndices y superíndices en varias líneas, 393. — Integrales múltiples, 394. — Construyendo operadores de tamaño variable, 394.	
5.11. Diagramas conmutativos con el paquete <i>amscd</i>	395
5.12. Teoremas y demostraciones	396
El paquete <code>amsthm</code> , 398.	
5.13. Parámetros globales en las fórmulas matemáticas	400
6. Internet	403
6.1. Hiperenlaces en <i>LATEX</i> con el paquete <i>hyperref</i>	403
Configuración de los enlaces, 405. — Configuración de los colores, 406. — Opciones de visualización específicas de PDF, 407. — El resumen de un documento PDF, 409. — Acceso a los menús de ACROBAT, 411. — Opciones adicionales, 411. — ¿Problemas con <i>hyperref</i> ?, 411.	

6.2. Incluyendo anotaciones en los documentos PDF con PDFLAT _E X	412
Notas de texto, 413. — Enlaces que realizan acciones, 414. — Las películas y los sonidos, 416.	
6.3. L _A T _E X y HTML. El sistema de conversión T _E X4ht	417
El proceso de conversión con T _E X4ht, 418. — Opciones del sistema T _E X4ht, 420. — Ficheros de configuración, 424. — Comandos y entornos básicos de T _E X4ht, 424.	
7. Presentaciones	433
7.1. Presentaciones con el paquete web	434
Idiomas, 434. — Visualización del documento PDF, 435. — Página del título, 436. — El «directorio» y el índice general, 437. — Filigranas y fondos, 438. — Barra de navegación, 440. — Panel de navegación, 441. — Otros comandos y opciones, 444.	
7.2. Una herramienta adicional para las presentaciones: PPOWER4	444
Los paquetes, 445. — Pausas: el paquete pause, 446. — Efectos de transición, 446. — Fondos: el paquete background, 447. — Enlaces con las primeras subpáginas, 448. — Los entresijos de PPOWER4, 449. — Asignando niveles a las porciones, 450. — Resaltando las porciones, 451. — Modos de trabajo, 452.	
7.3. Presentaciones con la clase prosper	452
Estilos de presentación, 453. — Opciones de la clase prosper, 455. — El proceso de compilación, 455. — El preámbulo de un documento prosper, 456. — Cuerpo de un documento prosper, 457. — Animaciones en pantalla, 459.	
8. Programación	463
8.1. Definiendo comandos de otro modo	463
Los comandos son como funciones, 464. — Delimitadores de los argumentos, 464. — Definiciones globales, 465. — Definiciones recursivas, 467. — El comando \let, 467.	
8.2. Contadores y longitudes en T _E X	468
8.3. Repitiendo un objeto	471
8.4. Configurando el diseño de los párrafos	473
Modificando el sangrado, 473. — Cuando los párrafos son muy especiales, 474.	
8.5. Sistematizando tareas	475
8.6. Controlando la situación	476
Algunos condicionales de T _E X, 477. — Algunos condicionales de L _A T _E X, 481. — Definición de nuevos condicionales, 482. — Los bucles, 483. — Más ejemplos, 485.	
Cómo hacerlo	489
Índice terminológico	503
Bibliografía	521

Índice de figuras

L*A***T***E*X básico en dieciocho lecciones

1.1.	Esquema básico de funcionamiento del compilador <i>T_EX</i>	4
1.2.	Esquema ampliado de funcionamiento de <i>L_AT_EX</i>	6
2.1.	Entornos de trabajo para distintos sistemas operativos	12
2.2.	¿Qué aspecto tiene un texto fuente para <i>L_AT_EX</i> ?	14
2.3.	Primera composición con <i>L_AT_EX</i>	15
2.4.	Segunda composición, una vez incluidos algunos paquetes útiles	18
2.5.	El preámbulo para nuestras lecciones	18
7.1.	Esquema del diseño de las páginas y sus parámetros	66
9.1.	Texto a dos columnas con figuras	86
15.1.	Ejemplo de bibliografía	122

Para ser un *L**A**T**E**X*perto

1.1.	Algunas de las 35 fuentes de Adobe	183
1.2.	Caracteres para Computer Modern Roman (codificados «OT1»y «T1»)	188
1.3.	Tablas de caracteres para Symbol	190
1.4.	Tablas de caracteres para marvosym	193
2.1.	Marcas con estilo de página headings en la clase book	212
2.2.	Esquema de las marcas con el paquete fancyhdr	218
2.3.	Marcas con el paquete fancyhdr y el estilo fancy	220
2.4.	Fragmento de base de datos para BIB <i>T_EX</i>	252
2.5.	Fragmento de un fichero BBL obtenido al ejecutar BIB <i>T_EX</i>	256
2.6.	Lista de algunos estilos para BIB <i>T_EX</i> que aparecen en CTAN	257
3.1.	Estructura general de una lista del entorno list	282
6.1.	Diferentes formas en las que se puede visualizar un documento PDF	407
6.2.	Esquema de funcionamiento de <i>T_EX4ht</i>	418
6.3.	Ventanas de ejecución y selección de opciones de TEXCONVERTER	420
7.1.	Índice general en el paquete web	438
7.2.	Barra de navegación personalizada en el paquete web	442
7.5.	Estructura de un documento <i>L_AT_EX</i> para la clase prosper	453
7.6.	Estilos de la clase prosper	454
7.7.	Construcción de una presentación con prosper	456
8.1.	Párrafo especial creado con el comando \parshape	474

Índice de cuadros

L**A****T****E**X básico en dieciocho lecciones

4.1.	Comandos para imprimir los signos correspondientes a los caracteres reservados	32
4.2.	Comillas y cómo obtenerlas directamente desde el teclado	36
4.3.	Guiones introducidos directamente desde el teclado	36
4.4.	Miscelánea de signos	38
5.1.	Nombre de los 68 colores predefinidos con la opción dvipsnames	46
6.1.	Jerarquía de las unidades de estructura	50
6.2.	Opciones básicas de las clases de documento	59
6.3.	Tamaños relativos de los tipos según la opción de la clase de documento	60
6.4.	Opciones de tamaño de papel y sus dimensiones	60
16.1.	Comandos para espaciados horizontales	134
16.2.	Símbolos matemáticos: letras griegas	134
16.3.	Nuevos nombres de algunos comandos en chemsym	136
17.1.	Unidades de longitud válidas en T <small>E</small> X	143

Para ser un LATEXpertó

1.1.	Idiomas de babel y sus opciones	166
1.2.	Traducción de los antetítulos a los idiomas del Estado español	171
1.3.	Paquetes para las 35 fuentes PostScript	187
2.1.	Definiciones de las marcas en el estilo headings	215
2.2.	Selectores de página y campo en el paquete fancyhdr	219
2.3.	Comandos del paquete achicago	266
3.1.	Controladores disponibles como opción para los paquetes graphicx y color	274
3.2.	Contadores y etiquetas para el entorno enumerate	277
3.3.	Etiquetas para el entorno itemize	280
3.4.	Especificadores de columna, separadores y argumentos con el paquete array	290
3.5.	Otros parámetros y comandos del entorno longtable	293
3.6.	Opciones del paquete caption2	306
4.1.	Comandos para las conexiones entre nodos	351
4.2.	Tipos de flechas en PStricks	353

4.3.	Puntos con PSTRicks	354
5.1.	Separación entre operadores matemáticos	359
5.2.	Símbolos ordinarios	361
5.3.	Símbolos ordinarios: fuente itálica de letras griegas mayúsculas	361
5.4.	Símbolos ordinarios: miscelánea	362
5.5.	Símbolos para operadores binarios	363
5.6.	Símbolos para operadores de relación	364
5.7.	Símbolos para operadores de relación: flechas	365
5.8.	Flechas verticales extensibles	365
5.9.	Símbolos para operadores de relación: miscelánea	366
5.10.	Símbolos extensibles no emparejados	367
5.11.	Delimitadores de apertura y cierre	367
5.12.	Efecto de los cambios de fuente en modo matemático	367
5.13.	Acentos en modo matemático	368
5.14.	Signos de puntuación	370
5.15.	Comandos para escalado manual de los delimitadores	378
5.16.	Nombres de funciones	389
5.17.	Funciones trigonométricas en castellano	390
5.18.	Operadores de acumulación de tamaño variable	393
6.2.	Modos de visualización en un documento PDF	408
6.3.	Efectos de transición entre páginas de un documento PDF	410
6.4.	Opciones de menú del programa ACROBAT	412
6.5.	Atributos comunes a todas las anotaciones de un documento PDF	414
7.1.	Controladores admitidos por el paquete web	434
7.2.	Estilos de las presentaciones con prosper	455
7.3.	Efectos de transición entre páginas con prosper	457

Parte I

L^AT_EX básico en dieciocho lecciones

Cómo funciona L^AT_EX



OBJETIVOS

- Comprender el esquema básico de funcionamiento de T_EX y L^AT_EX.
- Conocer las diferentes salidas que produce L^AT_EX.
- Aprender algunas características de los documentos PostScript y PDF.

ESTA lección, de carácter introductorio, describe el esquema de funcionamiento y los conceptos y herramientas básicos para escribir con L^AT_EX. Por estos motivos es imprescindible leerla con detenimiento y comprenderla en su totalidad. Ahora no es el momento de describir con detalle las herramientas ni de exponer todas las posibilidades que L^AT_EX nos ofrece; en las lecciones siguientes abordaremos estas cuestiones. Pero para poder sacarle el máximo provecho, es muy conveniente entender su filosofía y las líneas generales de funcionamiento del programa.

§1.1 Comprendiendo el funcionamiento

ANQUE T_EX es un programa de computador diseñado específicamente para escribir bellos textos, especialmente textos que incluyan fórmulas, no es un «editor de textos» en el sentido usual del concepto. T_EX es lo que podríamos llamar un «maquetador» o «compositor» de textos. Para comprender lo que esto significa, entender su «transportabilidad» entre las diferentes plataformas informáticas (una cualidad sumamente interesante) y poder actuar eficazmente sobre él, es necesario conocer su funcionamiento y los ingredientes básicos del mismo. En la figura 1.1 presentamos un esquema sobre el modo de actuar de T_EX y las tres etapas básicas (texto fuente, composición y visionado o impresión) que configuran el proceso.

Texto fuente. En cualquier plataforma informática, con cualquier editor de textos capaz de producir un fichero «sólo texto», se escribe un documento en el que, además del texto propiamente dicho, se introduce, de acuerdo con una determinada sintaxis, información sobre la estructura final que se desea para dicho texto. Supondremos almacenado este documento en un

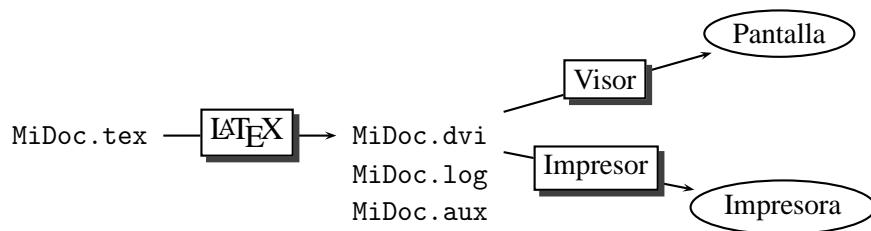


Figura 1.1: Esquema básico de funcionamiento del compilador T_EX

fichero de nombre *MiDoc.tex*, al que hemos asignado la extensión *.tex*, que si bien no es obligatoria, sí es la usual y aconsejable. Estos ficheros «sólo texto» son transportables entre sistemas informáticos, están perfectamente adaptados a Internet y pueden ser leídos y modificados por los diferentes editores.

Composición o compilación. El texto fuente se procesa para darle formato y componerlo. El sistema realiza la tarea que correspondería a un impresor de una imprenta Gutenberg, es decir, aprende del autor lo que éste quiere hacer y se ocupa de componer líneas y páginas, dar formato a capítulos, secciones, notas a pie, índice general, etc. La composición de nuestros documentos descansa en los siguientes dos elementos principales:

- El núcleo central T_EX, llamado compilador en argot informático, que es un programa capaz de ejecutar tareas que le son ordenadas mediante un «lenguaje» lo suficientemente rico y flexible para poder resolver casi cualquier situación que se pueda presentar en esta materia. El libro de Donald E. Knuth [32] describe este lenguaje y su utilización con todo detalle.

El contrapunto a esa flexibilidad es, en opinión de algunos, una cierta complejidad de manejo. Por ello, al poco tiempo de crearse la herramienta se comenzaron a desarrollar intérpretes con la finalidad de facilitar la comunicación del que escribe el texto fuente con el compilador T_EX.

- El formato L^AT_EX, creado por Leslie Lamport [37], es, en su actual versión, el intérprete más versátil porque, a sus ya importantes capacidades, ha incorporado las herramientas más destacadas del otro intérprete famoso, AMS-T_EX, desarrollado por la Sociedad Matemática Americana (American Mathematical Society) y descrito en el libro de Michael Spivak [62]. L^AT_EX, además de facilitar la comunicación con el motor central, tiene un enorme conocimiento estilístico sobre la forma de presentar un texto con la mejor calidad profesional y evita al autor del texto fuente tener que tomar decisiones sobre el formato preciso con el que quiere que aparezca su trabajo, permitiéndole concentrarse en el contenido del documento y las estructuras básicas del mismo.

A partir de este momento, y una vez distinguidos el compilador T_EX y el formato L^AT_EX, utilizaremos indistintamente ambos términos para referirnos al proceso de composición de un documento T_EX. Al compilar el texto fuente *MiDoc.tex* se produce un nuevo fichero, llamado *MiDoc.dvi*, que contiene toda la información «cajística» para imprimir el texto final, pero en el cual los tipos (las fuentes) podríamos decir que están todavía sin entintar,

son «blancos». Este fichero tiene también un cierto carácter de transportabilidad entre plataformas informáticas como ocurría con el fichero fuente; cualquier programa de visionado o impresión para \TeX ha de permitir obtener de él una copia en la pantalla del computador o en la impresora, utilizando la «tinta» adecuada, sin importar cuál sea la plataforma informática o los modelos de pantalla o impresora. La extensión `dvi` hace referencia a este carácter de independencia respecto del periférico (**device independent**).

Durante la actuación de \LaTeX se producen también otros ficheros que son herramientas auxiliares para conseguir el objetivo final (`MiDoc.aux`), o bien contienen información técnica sobre el propio proceso de compilación (`MiDoc.log`).

Visualización o impresión. Para poder ver en la pantalla de un computador e imprimir el resultado de la composición que \LaTeX ha realizado es necesario «entintar» los tipos del fichero `dvi` de salida. De ello se ocupan programas específicos diseñados con esa finalidad, de los que existe una gran variedad en cada plataforma informática. El mismo fichero (por ejemplo, `MiDoc.dvi`), y, por tanto, la misma composición, puede imprimirse (o verse en la pantalla del computador) con diferentes niveles de acabado, dependiendo de las capacidades de la impresora (o monitor) y de la «finura de entintado» o resolución que el programa de impresión (o visualización) utilice. Refiriéndonos por ejemplo a la impresión, el resultado que se obtiene en una modesta impresora de inyección de tinta de 300 puntos por pulgada es exactamente el mismo, salvo el nivel de resolución y precisión en el acabado, que el que se obtiene en la filmadora de una imprenta profesional con varios miles de puntos por pulgada, como la utilizada al imprimir este libro.

Pros y contras del sistema

El esquema descrito en el apartado anterior es muy simple y resulta muy atractivo. Si a eso añadimos que \LaTeX es un programa gratuito y de dominio público y que está implementado en las diferentes plataformas informáticas, el panorama es inmejorable: *un excelente impresor a nuestro servicio ¡que no nos envía facturas!* Efectivamente, ésa es la situación, aunque el esquema real de trabajo es un poco más complejo:

- En primer lugar, \LaTeX se ocupa de componer el manuscrito, pero para ello hay que indicarle (como se haría con un impresor), por ejemplo, dónde empiezan y cuáles son los títulos de capítulos, secciones, subsecciones, qué textos son notas a pie de página, etc. Esta información hay que proporcionársela en un «lenguaje» preciso que él entienda. Además del texto propiamente dicho se deben incluir instrucciones muy concretas que instruyan al programa sobre diversas características del formato. Aquí pueden producirse errores, que habrá que depurar y corregir realizando las oportunas modificaciones en el texto fuente.
- \LaTeX ha sido diseñado para que sea capaz de hacer algunas tareas de forma automática, como la confección de índices (general, de figuras, de cuadros, etc.), la realización de referencias cruzadas, la producción de las listas bibliográficas, etc. Para la realización de dichas tareas es por lo que se generan la mayoría de los archivos auxiliares, que se utilizarán en una segunda compilación.

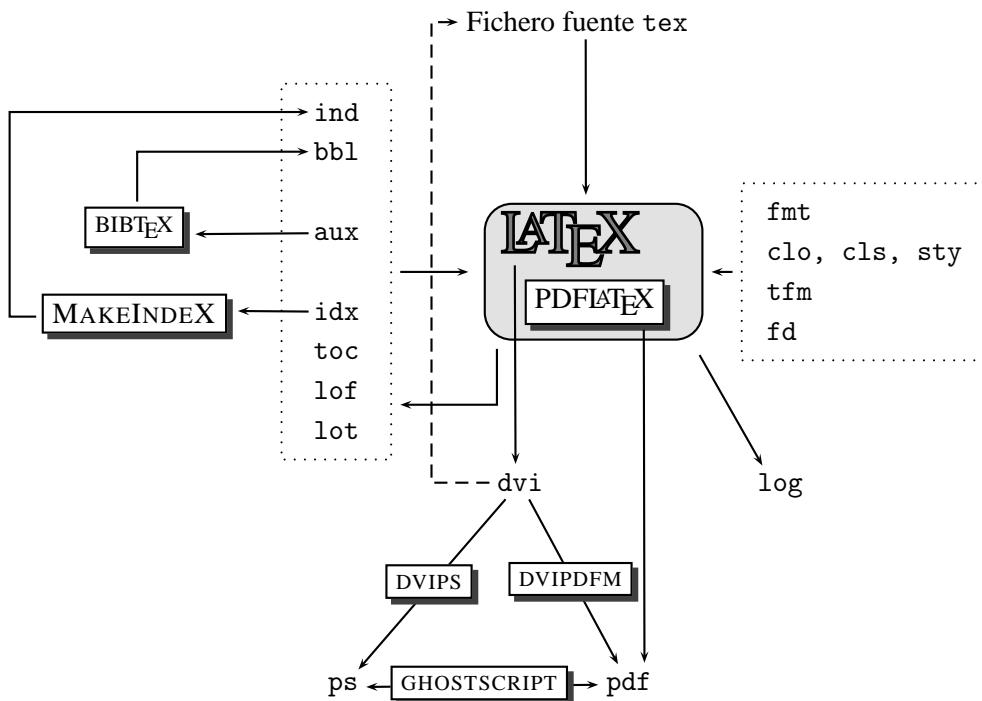


Figura 1.2: Esquema ampliado de funcionamiento de LATEX

- Finalmente, otra de las ventajas del sistema es la capacidad de producir, a partir del mismo texto fuente, documentos en diferentes formatos electrónicos, adaptados al uso que de ellos se pretenda hacer. Entre las principales salidas están los documentos PostScript, PDF y HTML, entre otros.

Resumiendo, si bien el esquema que aparece en la figura 1.1 es adecuado para describir la estructura del funcionamiento, el desarrollo en la práctica del quehacer no es tan lineal; la figura 1.2, con sus flechas de retroalimentación del texto fuente, refleja de forma más fiel el día a día del trabajo con LATEX. La «ida y vuelta» del texto fuente al documento DVI (o a los documentos PS o PDF) puede ser muy frecuente en los primeros manuscritos que se realicen, pero, como ocurre con cualquier proceso de aprendizaje, disminuye sensiblemente con la experiencia.

§1.2 Documentos en formato PostScript

COMO vemos en la figura 1.2, además de los archivos DVI podemos obtener como «salida» otros tipos de ficheros. Entre ellos están los ficheros PostScript, con extensión ps, que son ficheros de texto que describen, en un lenguaje denominado también PostScript, las páginas del documento. Su uso está ampliamente extendido, por ejemplo, entre los impresores profesionales. Entre las principales características de estos ficheros se encuentran las siguientes:

- Son independientes de la plataforma, en el sentido de que el aspecto del documento no variará de una impresora a otra. Lo que sí cambiará será la calidad con la que está impreso, pues obviamente no es lo mismo imprimir en una impresora de inyección de tinta de 200 ppp que hacerlo en una imprenta profesional de 2 400 ppp.
- La descripción de los elementos del documento es vectorial, es decir, puede ser escalada, sin perder calidad (para explicarlo mejor: un carácter no es una descripción punto a punto del mismo, sino un pequeño programa que lo describe).
- Permiten la inclusión de gráficos de alta calidad.

A partir de un fichero `tex`, más precisamente a partir del fichero `dvi` que su compilación produce, el programa DVIPS genera un fichero PostScript. El programa DVIPS, que se incluye en casi todas las distribuciones estándar de `TeX`, permite, además de esta conversión, muchas otras tareas: inclusión de gráficos (rotación y escalado, colores), enlaces de hipertexto (tanto a otras partes del documento como a direcciones de Internet), etc. Muchos de estos aspectos serán cubiertos en sucesivas lecciones.

Los ficheros PostScript, como ya hemos comentado, contienen un lenguaje de impresión y, por tanto, pueden ser copiados directamente a nuestra impresora, aunque, ¡cuidado!, para ello nuestra impresora debe entender el lenguaje PostScript. La mayor parte de las impresoras láser modernas lo entienden, mientras que no ocurre lo mismo con la mayoría de las impresoras de inyección de tinta (menos profesionales). ¿Significa eso que no podemos imprimir ficheros PostScript en una impresora personal que no es PostScript? En absoluto. Tenemos a nuestra disposición una herramienta que realiza el trabajo: el tandem GHOSTSCRIPT-GSVIEW, que son programas de libre distribución (existentes en numerosas plataformas, desde luego en MS-WINDOWS y LINUX, en éste último bajo el nombre de GHOSTVIEW) y que permiten visualizar en pantalla e imprimir, en cualquier impresora instalada en el sistema, los documentos PostScript.

§1.3 Documentos en formato PDF

EN los últimos años se está imponiendo en la publicación electrónica de documentos un nuevo formato: el PDF (Portable Document Format), el lenguaje natural de la familia de productos Adobe Acrobat. El principal objetivo de este nuevo formato es permitir que los usuarios puedan fácilmente intercambiar y manipular documentos electrónicos independientes del hardware y software con el que fueron creados. El lenguaje PDF es un heredero del lenguaje PostScript y, al igual que éste, tiene como una de sus principales características la independencia del dispositivo de salida y de la resolución.

El lenguaje PDF incorpora diversos objetos tales como marcadores o enlaces que, estrictamente hablando, no forman parte del documento, pero que son muy útiles para una visualización interactiva. Los ficheros PDF están construidos como una sucesión de objetos numerados, de forma similar a los ficheros PostScript. Sin embargo, los ficheros PDF no siguen las especificaciones del lenguaje PostScript, por lo que, al menos hoy por hoy, no pueden enviarse directamente a una impresora PostScript.

El formato PDF es hoy en día el preferido para publicar en Internet documentos electrónicos con un formato de alta calidad. Entre sus principales ventajas están las siguientes:

- PDF fue especialmente diseñado para crear documentos que debían ser distribuidos electrónicamente, mientras que PostScript no es más que un lenguaje para controlar las impresoras. En este sentido, PDF es mucho más versátil.
- PDF posee mecanismos de compresión interna, por lo que no es necesario el uso de herramientas externas de compresión y empaquetamiento. El tamaño electrónico de un documento PDF es notablemente menor que el equivalente en formato PostScript.
- PDF permite la inclusión de enlaces URL, manteniendo una buena calidad tipográfica, muy superior a la de los actuales formatos HTML.
- PDF permite la inclusión de gráficos sofisticados, como por ejemplo fotografías, en formato JPG muy compactado, mientras que los gráficos incluidos en los archivos PostScript incrementan enormemente el tamaño electrónico del documento.
- Los visores PDF y las herramientas de impresión, por ejemplo los programas GSVIEW, GHOSTVIEW y ACROBAT READER, se distribuyen libremente y son bastante amigables.

§1.4 Creando documentos PDF

PARA crear documentos PDF a partir de nuestros archivos TEX existen tres caminos básicos, que ya han sido presentados en la figura 1.2. La opción más clásica ha consistido en utilizar el siguiente camino:

tex \Rightarrow dvi \Rightarrow ps \Rightarrow pdf

La última etapa en este camino, es decir, la obtención de un fichero PDF a partir de un fichero PS, se puede cubrir, por ejemplo, con la ayuda del programa gratuito GHOSTSCRIPT, ejecutando la utilidad PS2PDF (lo que también es posible realizar desde un menú del programa GSVIEW) o utilizando el programa comercial ACROBAT.

Recientemente han surgido dos nuevas alternativas, más simples de utilizar y que producen resultados de la misma o superior calidad:

DVIPDFM: Es un programa que transforma los archivos DVI creados por L^AT_EX en archivos PDF; puede considerarse un programa dual del ya citado DVIPS.

PDFL^AT_EX: Es la **opción que recomendamos**, ya que el programa transforma el archivo TEX en un archivo PDF en un solo paso.

Los diferentes métodos de obtención de archivos PDF que hemos descrito no son equivalentes. Cada uno de los caminos tiene sus ventajas e inconvenientes, fundamentalmente en lo que se refiere a la inclusión de objetos gráficos, que se irán detallando a lo largo del libro.

PARA SABER MÁS

- ▶ Los documentos PostScript están pensados para ser directamente «copiados» en la impresora. Sin embargo, existe una versión especial, los gráficos PostScript encapsulados (usualmente con la extensión `eps`), que están pensados para ser incluidos en otros documentos. La forma de incorporar gráficos PostScript (y de otros tipos) en nuestros documentos se encuentra en la lección 9.
- ▶ Los documentos PDF son especialmente atractivos por la posibilidad de utilizar enlaces de hipertexto, incluir marcadores, realizar anotaciones electrónicas, etc. En la lección 8 encontrará más detalles.
- ▶ Si queremos producir documentos para colgar de una página web y no nos preocupa demasiado el aspecto estético de los mismos, quizás una buena alternativa sea producir documentos HTML. La conversión de nuestros archivos TEX en documentos HTML es realmente simple; analizaremos este problema en la sección II.6.3.

Primera cita: composición de un documento



OBJETIVOS

- Experimentar el procedimiento de composición de un texto con L^AT_EX.
- Editar, compilar y visualizar un texto.
- Establecer un preámbulo básico para los primeros documentos.
- Aprender a reaccionar ante posibles errores.

PAQUETES COMENTADOS: inputenc, fontenc, babel con opción spanish

VAMOS a seguir el esquema básico de funcionamiento de T_EX y L^AT_EX descrito en la primera lección para componer algunos textos sencillos. Antes de componer con L^AT_EX, enumeraremos los programas que debemos instalar en nuestro computador y describiremos las características generales de los entornos integrados diseñados específicamente para realizar todas las tareas necesarias para la escritura y composición de documentos L^AT_EX.

Aprovecharemos los distintos documentos procesados a lo largo de la lección para ir configurando un preámbulo que nos sirva de base en las siguientes lecciones. *En todas las lecciones supondremos que estamos utilizando este preámbulo estándar.*

También analizaremos algunos de los errores que se pueden producir al compilar y veremos la manera de corregirlos.

§2.1 Instalación de programas

ANQUE el contenido de este libro no está ligado a ningún sistema operativo o distribución particular de L^AT_EX, vamos a hacer una excepción en este apartado describiendo la secuencia de programas a instalar y los entornos habituales de trabajo para la composición de textos con L^AT_EX.

Vamos a necesitar tener instalados los programas que relacionamos a continuación:

- ACROBAT READER: programa gratuito de Adobe para visualizar e imprimir ficheros PDF.

- GHOSTSCRIPT, GSVIEW-GHOSTVIEW: programas gratuitos con licencia GPL¹ para procesar ficheros PostScript.
- Una distribución de TEX y LATEX. Recomendamos: MiKTEX para MS-WINDOWS, teTEX para LINUX/UNIX, y TEXShop o iTEXMac para MacOSX. Todos gratuitos con licencia GPL. En entornos Mac también se pueden instalar los sistemas de pago TEXtires y OzTEX.
- Un programa que integre en un mismo entorno los distintos procesos de la composición de documentos TEX, edición, composición, visualización e impresión. De entre los entornos de este tipo, recomendamos: TeXnicCenter para MS-WINDOWS, Kile para LINUX/UNIX, todos con licencia GPL. También recomendamos, como alternativa, el estupendo entorno WinEdt para MS-WINDOWS (con un precio muy bajo). En entornos Mac los sistemas antes citados ya incluyen entornos de trabajo.

En la figura 2.1 mostramos alguna de las ventanas de los programas citados.

Vamos a trabajar con LATEX utilizando uno de estos entornos integrados. Todos tienen en común las siguientes características que facilitan la tarea de editar y componer textos:

- Proporcionan una ventana de edición para nuestro documento fuente que tiene menús y barras de herramientas, con las facilidades habituales de edición, incluida la corrección ortográfica. Desde estos menús también se pueden incluir muchas de las declaraciones y órdenes de LATEX necesarias para la correcta composición de los textos.
- Muestran una ventana de compilación, donde se van anotando los distintos pasos, errores y mensajes de precaución generados en la compilación. Incluyen opciones de menú y botones en la barra de herramientas para localizar simultáneamente los errores en la ventana de edición y en la ventana de compilación. Esta forma de localización de errores es muy útil para la depuración de fallos.
- Permiten acceder a los programas de visionado y transformación de los ficheros generados por el compilador.
- Ofrecen la posibilidad de gestionar de forma razonable «proyectos» correspondientes a textos extensos (libros, memorias, tesis doctorales, etc.) en los que se puede trabajar con varios documentos fuente, gráficos, tablas, índices, bases de datos con bibliografía, etc.

 En el CD-ROM que acompaña al libro se incluyen los programas gratuitos recomendados, junto con un sencillo programa de instalación para los sistemas operativos de MS-WINDOWS. También contiene recursos suficientes para realizar los ejercicios de este libro.

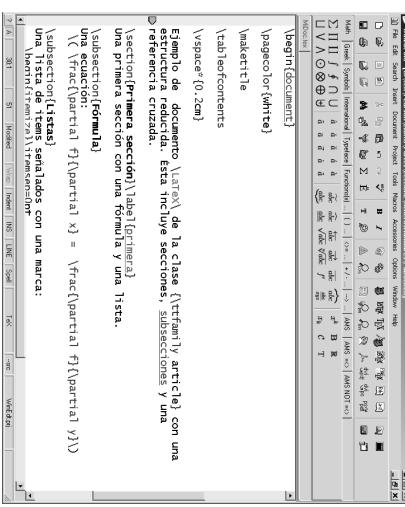
§2.2 ¿Qué contiene un documento fuente para LATEX?

UNA vez instalados los programas descritos en el apartado anterior, y antes de comprobar que todo funciona correctamente, vamos a analizar en qué consiste un texto fuente para LATEX. Para ello nos referiremos al documento que aparece en la figura 2.2. En el lado izquierdo de la

¹ «GNU General Public Licence», creada para garantizar la libertad de compartir y cambiar software «libre» con la seguridad de que es gratis para todos los usuarios.

Primera cita: composición de un documento

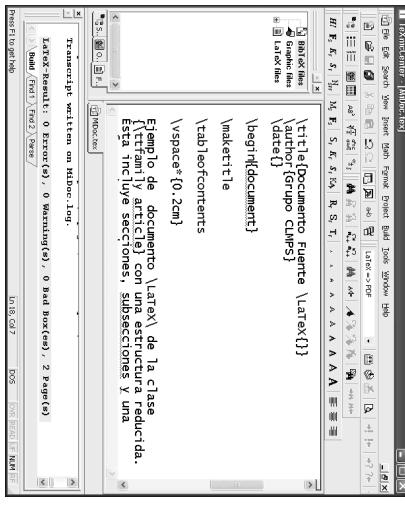
Acrobat Reader: Ver e imprimir documentos PDF.



Documento Fuente L^AT_EX

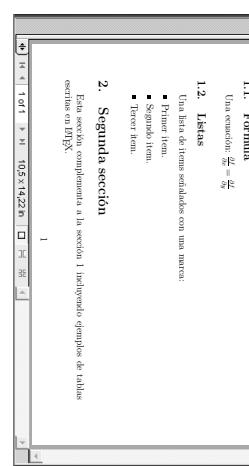
```
\begin{document}
\maketitle
\tableofcontents
\begin{center}
\textbf{\LARGE Ejemplo de documento LATEX de la clase \texttt{article} con una estructura reducida.\\
Est\'a incluye secciones, subsecciones y una \\ tabla de contenidos.\\
\vspace{0.2cm}}
\end{center}
\end{document}
```

TeXnicCenter: Compilar, ver, etc., de la opción «BUILD/CURRENT FILE».



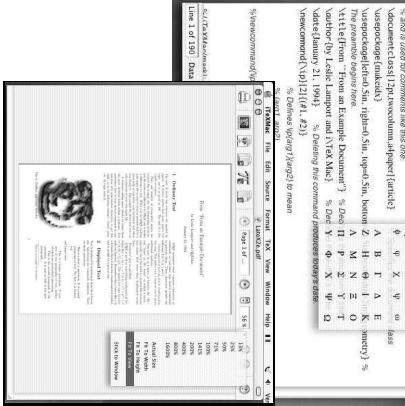
TeXnicCenter: Compilar, ver, etc., desde la opción «BUILD/CURRENT FILE».

Kile: Compilar, ver, etc., desde «HERRAMIENTAS» en el menú.



Kile: Compilar, ver, etc., desde «HERRAMIENTAS» en el menú.

ITEXMac: Editar, compilar y ver, en una misma ventana.



iTeXMac: Editar, compilar y ver, en una misma ventana.

Figura 2.1: Entornos de trabajo para distintos sistemas operativos. Integran los distintos procesos sobre un documento fuente L^AT_EX (edición, compilación, visionado, impresión, etc.) en un solo programa

figura hemos reproducido el contenido de un texto fuente que produjo, tras compilarlo dos veces con L^AT_EX, el resultado que aparece en el lado derecho de la figura.

El documento de la citada figura contiene una serie de declaraciones y órdenes para el compilador, que a partir de ahora llamaremos «comandos». La sintaxis de cada uno de estos comandos se corresponde con una palabra precedida por la «antibarra» ‘\’ y puede ir seguida de algunos «argumentos» delimitados por llaves o corchetes. El documento también contiene un «entorno», que es una orden que afecta sólo a una parte del documento, bien delimitada por las declaraciones \begin{itemize} y \end{itemize}.

Otra observación importante sobre cualquier texto fuente para L^AT_EX es la división, que hemos remarcado en la figura, entre una primera parte, que llamaremos «preámbulo» del documento fuente, en la que se hacen declaraciones y se dan órdenes que van a afectar a todo el documento, y una segunda parte que es la que contendrá el texto propiamente dicho del documento y que llamaremos «cuerpo» del documento fuente; ésta última, que es en realidad un entorno, siempre se inicia con la declaración \begin{document} y termina con \end{document}.

Como podemos ver en la salida producida, L^AT_EX ha elegido el tipo de letra, el ancho del texto, ha justificado el texto por la derecha, ha numerado las páginas de un determinado modo, ha interpretado una fórmula y una lista, ha entendido la estructura establecida por algunas declaraciones de sección y subsección y ha creado el índice general. Y todo eso lo realiza con independencia de que las líneas en el fichero fuente sean más o menos cortas, estén o no justificadas, las palabras estén separadas por un único espacio o por varios...

Observe que L^AT_EX está programado para dar formato al texto de acuerdo con unos parámetros internos predefinidos que dependen de la clase de documento de que se trate y pueden ser diferentes según se trate de un artículo, un libro... Naturalmente todos estos parámetros se pueden cambiar para personalizar un documento, pero en primera instancia es preferible dejarle hacer a él, porque L^AT_EX ha sido diseñado para eso, para ocuparse por sí mismo de dar formato a los textos, con un nivel de buen gusto y calidad equiparables a los que se obtendrían en una buena imprenta.

Conforme vayamos avanzando en las siguientes lecciones iremos viendo lo que significa cada uno de los comandos usados en este texto fuente; no podríamos continuar, sin embargo, sin analizar dos comandos que tienen mucho que ver con los verdaderos protagonistas de todo el libro: T_EX y L^AT_EX. Los logotipos L^AT_EX (incluido en el documento de la figura 2.2) y T_EX se obtienen con las órdenes dadas por los comandos

\LaTeX	\TeX
--------	------

Estos comandos no tienen ningún argumento, sólo imprimen el logotipo. Observe que hemos escrito \LaTeX\ , añadiendo una barra inclinada y un espacio; de haber escrito solamente \LaTeX, el compilador no habría insertado un espacio entre el logotipo y la siguiente palabra.

§2.3 Componiendo el primer documento



EN este apartado llevaremos a cabo la primera experiencia con L^AT_EX. Podremos comprobar, en primer lugar, si nuestra instalación se ha realizado correctamente y nos iremos familiarizando con la creación de un texto fuente. Desde la primera experiencia aparecerán problemas, fallos en

Primera cita: composición de un documento

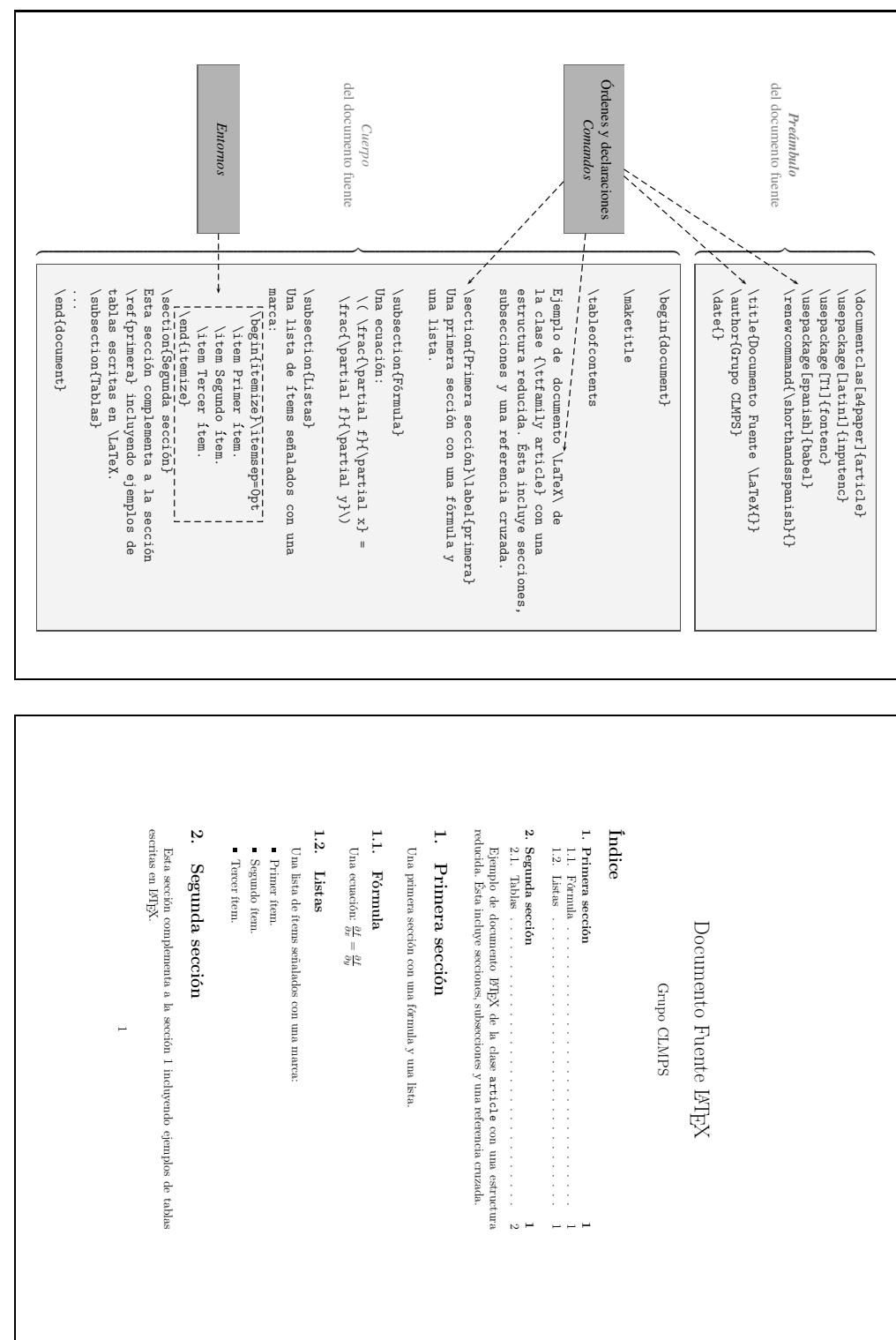


Figura 2.2: ¿Qué aspecto tiene un texto fuente para L^AT_EX? En el lado izquierdo aparece el contenido de un documento fuente y en el derecho el documento compuesto por L^AT_EX

el documento de salida. La intención al presentarlos así no es molestarle; sólo deseamos llamar la atención sobre la trascendencia del preámbulo del documento fuente. De hecho, el proceso que seguiremos nos conducirá a un preámbulo adecuado para todas las lecciones de esta primera parte, preámbulo que resumiremos en el apartado siguiente.

Comenzamos editando un nuevo documento con las líneas:

```
\documentclass{article} \begin{document}  
Saludos desde \LaTeX. Haciendo una nueva compilación de texto,  
con demasiada impaciencia por ver si todo funciona.
```

```
Escrito y compilado el día \today  
\end{document}
```

que guardaremos en un fichero, por ejemplo de nombre `MiDoc.tex`².

En el momento de compilar tendremos que elegir el tipo de salida que queremos obtener. En principio el compilador \LaTeX nos proporciona como salida un fichero DVI, que con el nombre sugerido antes daría lugar al fichero `MiDoc.dvi`. Tal y como presentamos en la lección anterior, también es posible obtener los ficheros `MiDoc.ps` y `MiDoc.pdf`, bien ejecutando los programas DVIPS o DVIPDFM sobre el fichero `MiDoc.dvi`, o bien compilando con PDF \LaTeX el documento fuente para obtener directamente el fichero `MiDoc.pdf`. Una vez elegido el tipo de fichero que queremos obtener, y dependiendo del entorno de trabajo que hayamos instalado, bastará con hacer «click» sobre las opciones del menú o los botones de la barra de herramientas correspondientes para compilar. Elegida una salida en formato DVI, \LaTeX genera el fichero `MiDoc.dvi` sin enviar ningún mensaje de error. Pulsamos en la opción «Ver» y lo veremos como en la figura 2.3.

Saludos desde \LaTeX . Haciendo una nueva compilación de texto, con demasiada impaciencia por ver si todo funciona.
Escrito y compilado el da March 9, 2003

Figura 2.3: Primera composición con \LaTeX

\LaTeX ha compuesto el texto interpretando nuestro manuscrito sin detectar ningún fallo. Sin embargo, nosotros sí observamos algunos «problemillas»:

- Las letras acentuadas han desaparecido.
- \LaTeX ha dividido la palabra «demasiada» para pasar a una nueva línea de texto, pero no ha seguido las reglas de «división silábica» del español.
- \LaTeX ha interpretado la palabra `\today` como la orden para poner la fecha del día de hoy, pero la ha escrito en inglés.

²Antes de componer el documento debemos darle el nombre guardándolo en un fichero de texto. Si olvidamos hacerlo y pulsamos en la opción de componer, nos podemos encontrar con un mensaje de error en la ventana del estado de la compilación. Esta situación se puede producir en algunos entornos de trabajo, como en TeXnicCenter.

Es fácil imaginar el origen de estos problemas: *LATEX normalmente trabaja en inglés, que no usa letras acentuadas y tiene unas reglas de división silábica distintas de las del español*. Por esta razón, LATEX, que conoce más de 150 idiomas y es capaz de utilizar cualquier alfabeto (con acentos y con caracteres mucho más extraños), necesita que le digamos el idioma y el alfabeto en el que estamos escribiendo nuestro texto.

Para pasarle éstas y otras informaciones a LATEX se utilizan declaraciones dadas mediante comandos. Normalmente, para tareas complejas, disponemos de unos ficheros denominados paquetes donde están reunidos los correspondientes comandos. Tras la instalación de LATEX dispondremos de muchos paquetes en nuestra máquina³. Después, sólo con una declaración, haremos que LATEX reciba todas las órdenes correspondientes al paquete.

La declaración que permite incluir un paquete es:

```
\usepackage [Opciones] {Paquete}
```

en la que se pueden incluir, junto al nombre del *Paquete* y entre corchetes, algunas *Opciones*. Estas declaraciones se deben incluir necesariamente en el preámbulo del documento, tal y como detallamos a continuación.

Preámbulo del documento fuente

Como ya decíamos en el apartado anterior, se llama «preámbulo» del documento a la parte del documento fuente que precede a la orden de inicio del texto ordinario: \begin{document}, y «cuerpo» a la parte comprendida entre \begin{document} y la orden \end{document} con la que finaliza el texto que el compilador considerará (véase la figura 2.2). De hecho, todo lo que siga a \end{document} será ignorado.

En el «preámbulo» se deben escribir todas las declaraciones que afectan a todo el documento y su primera línea debe ser la declaración de la clase de documento.

Para nuestros primeros documentos, en los que trabajamos con textos cortos, hemos elegido, entre las disponibles, la clase de documento article. Además, junto a la declaración de la clase de documento especificamos opciones para el tamaño del papel a utilizar y el tamaño de los tipos (fuentes). Así, para utilizar hojas DIN A4 y tipos de 11 puntos, escribimos:

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

También hay que incluir en el preámbulo las declaraciones de los paquetes que queremos utilizar. A continuación vamos a modificar el preámbulo de MiDoc.tex con las declaraciones de uso de los paquetes que resuelven los problemas detectados. El comando

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

declara el uso del paquete inputenc con la opción latin1. Con este paquete y esta opción le decimos a LATEX que estamos usando la codificación europea del teclado de nuestro computador, válida, por ejemplo, con los sistemas operativos MS-WINDOWS o LINUX. Ahora LATEX reconoce las vocales acentuadas y los demás símbolos del teclado, con lo que evitamos su desaparición del texto.

³Si realizó la instalación desde el CD-ROM, tendrá a su disposición todos los paquetes citados en este libro.

La siguiente línea declara el uso del paquete `fontenc` con la opción `T1`:

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

Su uso no es imprescindible y, a diferencia del paquete anterior, no está relacionado con la introducción de caracteres con tildes desde el teclado en un documento, sino con la gestión interna que `TeX` realiza con las fuentes o «tipos» para producir la salida de tales caracteres acentuados en el documento compilado. Para explicar esto hemos de referirnos a que Donald E. Knuth, además de `TeX`, creó otro compilador, llamado `METAFONT`, capaz de generar los tipos que `TeX` utilizaba. En lugar de crear un tipo para, por ejemplo, el carácter ‘a’, otros para ‘á’, ‘à’, etc., optó por crear un tipo para el carácter ‘a’ y otro para la tilde aguda ‘’’, que puede ser utilizado en diferentes letras. Durante el proceso de compilación `TeX` produce un tipo como el ‘á’, que no está disponible, superponiendo al tipo ‘a’ la tilde aguda. En el caso de disponer de una fuente de tipos en la que los caracteres acentuados existen como tales, no es necesario realizar el proceso de superposición anterior, lo que evita algunos errores de división silábica automática en palabras acentuadas. El paquete `fontenc` con la opción `T1` sirve para indicar a `TeX` que disponemos de fuentes de tipos con caracteres acentuados (entre otras cosas) y que utilice éstos en lugar de construirlos. Tiempo atrás eso no era posible con todos los formatos de salida, pero actualmente si instalamos en nuestro computador el sistema `cm-super`⁴, tendremos a nuestra disposición una extensión de las fuentes creadas por Knuth, adecuadas a los diferentes formatos de salida, que incorporan los caracteres acentuados y con la que la necesidad de una división silábica manual resulta absolutamente excepcional.

Para terminar, declararemos el uso del paquete `babel` con la opción `spanish`

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

Esto le dice a `LATeX` que vamos a escribir en español. Así, `LATeX` se guiará por las reglas del español para la división silábica, e interpretará en nuestro idioma algunos comandos como el utilizado para poner la fecha del día, `\today`.

Una vez incluidas las declaraciones de los paquetes nuestro documento `MiDoc.tex` queda con el siguiente contenido:

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[spanish]{babel}
\begin{document}
```

Saludos desde `\LaTeX`. Haciendo una nueva compilación de texto,
con demasiada impaciencia por ver si todo funciona.

```
Escrito y compilado el día \today
\end{document}
```

⁴Si se realiza la instalación desde el CD-ROM y al instalar el sistema `MiKTeX` optamos por la instalación completa, se instalan estas fuentes. Si sólo hacemos la instalación reducida, deberemos instalar el paquete desde el gestor de paquetes de `MiKTeX`.

Ahora, después de volver a compilar nuestro fichero, en la figura 2.4 podemos observar que han desaparecido los «problemillas» de la primera compilación.

Saludos desde L^AT_EX. Haciendo una nueva compilación de texto, con demasiada impaciencia por ver si todo funciona.

Escrito y compilado el día 18 de marzo de 2003

Figura 2.4: Segunda composición, una vez incluidos algunos paquetes útiles

Conviene que este preámbulo sea completado con la siguiente declaración adicional que interpretaremos mucho más adelante, en la segunda parte del libro, concretamente en la sección II.1.3:

`\renewcommand{\shorthandsspanish}{}{}`

Observe que entre las últimas llaves no hay nada, ni siquiera un espacio. No importa si no entiende esta sentencia, por otra parte es normal, ¡no la hemos explicado!, pero su inclusión nos ahorrará un buen montón de sorpresas desagradables.

§2.4 El preámbulo para nuestras lecciones

TA^L y como hemos anunciado antes, con los pasos del último apartado conseguimos un preámbulo mínimo, coherente y que evita que salgan a la luz algunos «problemas internos» de L^AT_EX. Supondremos a lo largo de todas las lecciones de esta primera parte que este preámbulo, mostrado en la figura 2.5, está cargado, lo que garantizará que la salida obtenida si reproduce el código fuente de los ejemplos es idéntica a la que ellos muestran. Las modificaciones posibles de este preámbulo se irán introduciendo poco a poco y consistirán fundamentalmente en el cambio de clase de documento (`book` por `article`) y la introducción de nuevos paquetes.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[spanish]{babel}
\renewcommand{\shorthandsspanish}{}{}
```

Figura 2.5: El preámbulo para nuestras lecciones

§2.5 Manejando algunos errores

VAMOS a terminar esta lección con unas anotaciones sobre el proceso de compilación y la depuración de errores. La compilación de un documento con L^AT_EX normalmente se puede realizar de una de las dos maneras que describimos a continuación. Usualmente, al instalar nuestro

entorno de trabajo, éste tendrá preseleccionada una de ellas y, seguramente, se podrá configurar para seleccionar la otra.

- **Compilación sin interrupciones** («non-stop mode»). L^AT_EX realizará la compilación del texto MiDoc.tex sin realizar «paradas», guardando en el fichero MiDoc.log la historia de la compilación, con especial atención a los errores cometidos. Cuando ocurra un error anotará en este fichero el número de la línea del fichero MiDoc.tex donde apareció, junto con una breve descripción del tipo de error. Una vez terminada la compilación podemos acudir a este fichero MiDoc.log para analizar los errores encontrados y corregir el documento fuente. En los entornos integrados de trabajo disponemos de herramientas que permiten hacer la depuración de errores de una forma razonable, localizándolos simultáneamente en el fichero MiDoc.tex y MiDoc.log. También ofrecen botones y opciones de menú para abandonar el trabajo de L^AT_EX si apreciamos que ha entrado en un bucle sin final aparente.
- **Compilación interactiva.** L^AT_EX inicia la compilación y va componiendo el texto deteniéndose cada vez que se produce un error; entonces nos mandará un mensaje de error en la ventana del sistema y se detendrá esperando una orden nuestra para continuar. En ese momento podremos optar por abandonar el trabajo, pulsando la tecla ‘x’, parar la compilación y volver al editor en una línea próxima a donde está el error, sólo con algunos editores, pulsando la tecla ‘e’, decirle que intente seguir, pulsando la tecla ‘Intro’, que intente terminar compilando lo que pueda, pulsando la tecla ‘s’ y, si nada de esto funciona o entra en un bucle sin fin, abandonaremos la compilación pulsando las teclas ‘Crtl-C’. En el modo interactivo, L^AT_EX también guarda en el fichero MiDoc.log toda la historia de la compilación, que deberemos consultar para depurar los errores existentes; es posible también que su entorno le ofrezca botones para buscar estos errores.

Pero, ¿con qué tipo de errores vamos a tener que enfrentarnos?, ¿va a ser difícil depurarlos? Esencialmente, los errores que se suelen producir al compilar un documento fuente con L^AT_EX pertenecen a alguno de los tres grupos siguientes.

- **Errores de escritura.** Son los que producimos al teclear nuestro documento. Por ejemplo, si tecleamos \lat e en lugar del comando \LaTeX le estamos enviando una orden al compilador que éste no entiende; entonces nos devolverá un mensaje con el número de la línea y un trozo de texto junto al falso comando (destacado por un salto de línea), advirtiéndonos que este comando no está definido:

! Undefined control sequence.

1.14 si tecleamos \lat e

en lugar ...

Bastará con reescribir bien la orden de esta línea 14, para tener depurado el error. Otro tipo de error de escritura surge cuando utilizamos equivocadamente alguna de las letras que por sí solas son comandos para L^AT_EX, como, por ejemplo, \, %, {, } o \$. En la lección 4 estudiaremos la lista de todos estos caracteres reservados, así como la manera de conseguir escribirlos en nuestro texto.

- **Errores de posición.** Son los que se producen cuando L^AT_EX encuentra un objeto inesperado en el trabajo que está componiendo. Un ejemplo de este tipo de error sucede cuando intentamos escribir parte del texto antes de la declaración `\begin{document}`. Por ejemplo, la palabra ‘Hola’ antes de que empiece el documento produce

```
! LaTeX Error: Missing \begin{document}.
See the LaTeX manual or LaTeX Companion for explanation.
Type H <return> for immediate help.
1.8 H
      ola
```

Este tipo de error puede ser involuntario, en el sentido de que puede estar provocado por un error de escritura en el preámbulo del documento.

También podemos provocar este tipo de errores cuando L^AT_EX está componiendo una fórmula y recibe alguna orden que no tiene sentido en este modo especial de trabajo (véase §16.1). Por ejemplo, si escribimos en nuestro documento fuente la orden `\LaTeX` recibiremos el mensaje de error

```
! You can't use '\spacefactor' in math mode.
\@-\spacefactor
\@m
1.12 $\$
```

- **Errores por omisión.** Éstos son los errores que provocamos cuando olvidamos dar la orden de finalizar una tarea. Los más habituales suceden al olvidar alguna llave { o }, o algún símbolo \$ en la declaración de una fórmula. Recibiremos mensajes como

```
! Missing $ inserted.
<inserted text>
      $
! Too many }'s.
```

Estos errores no suelen ser graves. Normalmente podemos detectarlos con sólo visualizar el documento compuesto. Pero podría ocurrir que alguno de estos errores fuese lo suficientemente grave para producir una parada de emergencia sin llegar a componer el documento. Estos casos más graves suelen deberse a que el documento fuente finaliza de forma imprevista para L^AT_EX, porque está a medio realizar una tarea en la composición. El caso más frecuente y también el más fácil de solucionar es el que produce el olvido de la última declaración: `\end{document}`.

La declaración `\end{document}` es necesaria para la composición del texto porque es la que ordena el final del trabajo: guarda el trabajo compilado y almacena la información que puede ser necesaria para la gestión automática de referencias cruzadas, índices, bibliografía, etc., en posteriores compilaciones.

Al compilar el documento `MiDoc2.tex`, que finalizó sin `\end{document}`, L^AT_EX detiene la compilación, y si estamos compilando sin interrupciones recibimos el mensaje:

```
! Emergency stop.  
<*>...teI/Leccion2/ficherosAuxiliares/MiDoc2.tex
```

No pages of output.

Transcript written on MiDoc2.log

Tendremos que acudir al fichero MiDoc2.log en busca de más información, y encontraremos en sus últimas líneas un mensaje que advierte de la falta de un final válido en el documento fuente.

Por el contrario, si L^AT_EX compila de manera interactiva, en la primera parada encontramos un asterisco «*», y si intentamos avanzar pulsando la tecla «Intro» obtenemos el mensaje:

*(Please type a command or say ‘\end’)

Para evitar estos problemas, una buena táctica es escribir el inicio y el final de los objetos de L^AT_EX de forma simultánea, antes de escribir su contenido. Por ejemplo, podemos escribir \begin{document} y unas líneas en blanco, e inmediatamente escribir \end{document} volviendo hacia arriba en el documento para empezar a escribir el texto.

Como última recomendación señalaremos que cuanto más corto es el texto que compilamos, más fácil resulta detectar y corregir los errores que pueda contener. Por esta razón es recomendable ir compilando nuestro documento conforme lo vamos escribiendo (una o dos veces por página) y corregir los errores que se vayan produciendo.

PARA SABER MÁS

-  En esta lección hemos utilizado unos «paquetes» que nos han resuelto los problemas planteados. En la actualidad existe toda una infinidad de paquetes a nuestra disposición para abordar una gran cantidad de situaciones. En el servidor de distribución de ficheros de T_EX (www.ctan.org), al que nos referiremos brevemente como CTAN, o desde cualquiera de sus «mirrors», así como desde los servidores de los distintos grupos de usuarios de T_EX (www.tug.org) encontrará una lista de direcciones de Internet donde es posible encontrar información sobre todos los paquetes y descargarlos en nuestro computador. Consulte también a este respecto el archivo TeXEnInternet.pdf, incluido en el CD-ROM. Los compiladores de T_EX más extendidos actualmente utilizan una base de datos que contiene la lista completa de los paquetes que el usuario ha instalado en su sistema. Si utilizamos uno de estos compiladores, cada vez que incorporemos un nuevo paquete en nuestro sistema necesitaremos realizar un proceso de actualización de dicha base de datos. La ejecución de este proceso depende de la distribución de T_EX que tengamos instalada; algunos de ellos ofrecen sistemas amigables (menús, botones) para llevar a cabo esta tarea.

Párrafos: alineación y párrafos especiales



OBJETIVOS

- Crear párrafos.
- Párrafos alineados a izquierda, a derecha, o centrados.
- Párrafos especiales: para citas textuales o poemas.
- Controlar la interlínea.

PAQUETES COMENTADOS: `indentfirst`, `doublespace`

INTRODUCIR un espacio o «blanco» entre palabras es, sin duda, la primera de las tareas que se presentan al escribir un texto. Casi inmediatamente aparece también la tarea de dividir el texto en párrafos. En la mayor parte de los «editores al uso» un espacio entre palabras no es más que eso mismo en el documento final impreso: un espacio entre palabras; dos espacios seguidos se transforman en dos espacios, que producen un blanco demasiado grande entre dos palabras, tres espacios dan un blanco aún mayor... y ¡es tan fácil pulsar dos o tres veces seguidas la barra espaciadora sin intención! Con los párrafos suele ocurrir algo similar: dos líneas blancas seguidas producen un espacio de separación entre párrafos consecutivos, tres líneas blancas un espacio mayor. Con L^AT_EX todo es muy distinto, nada es tan sensible a uno o dos blancos involuntarios; uno, dos, tres o ¡cincuenta! espacios, en L^AT_EX hacen sólo un blanco normal entre palabras. Esto es sólo una muestra de la gran independencia entre la forma en que se escribe el documento fuente y el resultado final.

§3.1 Párrafos, sangría y saltos de línea

MIENTRAS que en la mayoría de los editores de texto más extendidos un «retorno de carro» supone el inicio de una nueva línea, L^AT_EX interpreta un salto de línea en el fichero fuente como idéntico a un espacio y no como el inicio de un nuevo párrafo o línea. En L^AT_EX para iniciar un nuevo párrafo ¡hay que dejar una línea totalmente en blanco!, es decir, pulsar al menos dos

veces la tecla correspondiente al retorno de carro. Dejar dos líneas blancas, tres, cuatro... es lo mismo que dejar una única. Como alternativa puede usarse el comando

```
\par
```

En documentos con una presentación cuidada es habitual que después de cada punto y aparte haya un sangrado (la línea que sigue a un punto y aparte comienza un poco desplazada hacia la derecha). L^AT_EX realiza esa tarea de forma automática, salvo en situaciones especiales (una de las cuales aparece señalada un poco más adelante). Si se desea evitar la sangría de un párrafo determinado hay que iniciar dicho párrafo con el comando

```
\noindent
```

A diferencia de lo que ocurre en español, en las lenguas anglosajonas los párrafos que van precedidos del título de un capítulo, sección... no llevan sangría. L^AT_EX respeta esta costumbre, a menos que esté cargado el paquete babel con la opción spanish (lo cual estamos suponiendo en todas estas lecciones, véase la lección 2), en cuyo caso la sangría se aplica también a dichos párrafos. Si el paquete babel no está cargado, puede conseguirse ese mismo comportamiento cargando el paquete indentfirst mediante la instrucción

```
\usepackage{indentfirst}
```

incluida en el preámbulo del documento fuente.

L^AT_EX controla los tamaños de la sangría y de la separación entre párrafos mediante sendas longitudes:

```
\parindent
```

```
\parskip
```

Éstas son las primeras longitudes de L^AT_EX que aparecen en este libro. Componer textos cuidadosamente supone, necesariamente, manipular una gran cantidad de longitudes, de las que depende toda la composición. No debe resultar extraño, pues, que tan rápidamente hagan su aparición en estas lecciones. La manipulación de las longitudes de L^AT_EX será ampliamente abordada en la lección 17; en el camino hasta ella irán apareciendo más y más longitudes. Por ahora nos contaremos con saber que una forma de asignar o cambiar el valor de una longitud es la que se ilustra en cualquiera de las líneas del siguiente código:

```
\parindent=1.5cm
```

```
\parskip=5mm
```

L^AT_EX siempre realiza la alineación del texto ordinario tanto por la izquierda como por la derecha (justifica por la derecha). Si se desea iniciar una nueva línea sin completar la actual, la forma de indicárselo es poner uno de los siguientes comandos:

```
\newline
```

```
\\\ [Salto]
```

```
\\"
```

```
\\\*
```

El parámetro optativo *Salto* es un número con su unidad (por ejemplo, 2.5cm, 7mm) e indica la distancia vertical adicional a la que debe situarse la nueva línea. En caso de que la nueva línea no quepa en la misma página, se iniciará una nueva página y el *Salto* se reabsorberá. La versión *

es idéntica a `\ \`, salvo que le indica a L^AT_EX que la línea que se inicia después del comando no debe pasar a la página siguiente. También se pueden combinar ambos efectos con `\ *[Salto]`.

Puesto que para L^AT_EX es indiferente que entre dos palabras del archivo fuente se haya dejado un espacio en blanco o más de uno, si por alguna razón se desea dejar un espacio de separación mayor entre dos palabras puede utilizarse el siguiente comando:

`\u`

con el que representamos el carácter `\` seguido de un espacio. Este comando corresponde al espacio «normal» entre dos palabras. También se utiliza como espacio de separación entre un comando y la palabra inmediatamente posterior, ya que los espacios en blanco después de un comando se ignoran (véanse el ejercicio 3.2 y el apartado 10.2).

EJEMPLO 3.1

Hacer un punto y aparte es muy fácil.
Pero no basta cambiar de línea.

Hay que dejar una en blanco.

Para iniciar `\ \` otra no es necesario
cambiar de línea. `\par`
Dejar uno o más espacios es irrelevante.
Si se desea ... continuación `\ \ \ \ \` es
necesario incluir un comando que lo genere.

Hacer un punto y aparte es muy fácil. Pero no basta cam-
biar de línea.

Hay que dejar una en blanco.

Para iniciar
otra no es necesario cambiar de línea.

Dejar uno o más espacios es irrelevante. Si se desea for-
zar un espacio como el que viene a continuación es necesario
incluir un comando que lo genere.

Iniciar párrafos y líneas. Recuerde que en todos los ejemplos de las dieciocho lecciones se supone, como mínimo, el preámbulo estándar presentado en el apartado 2.4

§3.2 División silábica

COMO la longitud de cada línea de texto es fija, constantemente se producen situaciones en las que la inclusión de una nueva palabra supera la longitud de la línea, pero excluirla hace que la correspondiente línea sea más corta. Para conseguir que el texto aparezca justificado por la derecha, sólo hay dos posibilidades: introducir espacios de separación adicionales entre algunas palabras o bien dividir por un sitio adecuado la última palabra de la línea introduciendo un guión. El compilador T_EX utiliza este último procedimiento y reparte el espacio sobrante (si existe) de forma uniforme entre todas las palabras de la línea.

Para poner los guiones L^AT_EX dispone de un algoritmo de división silábica, que puede ser diferente según los idiomas. Estos algoritmos funcionan esencialmente bien y sólo muy ocasionalmente el resultado que el algoritmo produce no es el adecuado. La forma de actuar del algoritmo se puede modificar para una determinada palabra indicándole posibles lugares en que puede introducir opcionalmente una división en dicha palabra. Esto se lleva a cabo mediante el comando

`\-`

en el lugar (o lugares) en los que esté permitido introducir un guión, con lo cual se consigue lo siguiente: en caso de que sea necesario dividir la palabra, L^AT_EX la dividirá por alguno de los sitios

que se le indican. Ese método tiene el inconveniente de que hay que buscar la posición en la que se encuentra dicha palabra en el fichero fuente y ponerle los guiones.

El siguiente comando proporciona un segundo método para indicar al compilador, en el preámbulo del documento fuente, el modo de dividir en sílabas las palabras que no divide de forma adecuada:

```
\hyphenation{palabra1,palabra2,palabra3...}
```

Por ejemplo, podemos incluir, en el preámbulo del documento, la línea:

```
\hyphenation{ma-ter-ni-dad,al-bo-ro-to}
```

Este procedimiento tiene la desventaja de que consume memoria durante el proceso de compilación y, por tanto, puede ralentizarlo, aunque esto es un problema menor, dada la velocidad de los computadores actuales.

Afortunadamente, sólo en contadas ocasiones es necesario acudir a alguna de estas soluciones porque, actualmente, existen algoritmos eficientes de división silábica en diferentes lenguas, el español entre ellas. El paquete *babel* (véase el apartado 2.3) selecciona el que corresponde a la división silábica de cada lengua.

Cuando L^AT_EX no puede conseguir llenar la línea perfectamente, bien sea por exceso, bien por defecto, emite un mensaje durante el proceso de compilación del tipo

```
Overfull \hbox (45.45274pt too wide) in paragraph at lines ...
Underfull \hbox (badness 10000) detected at line ...
```

Se trata de mensajes de carácter informativo que ocasionalmente pueden ser importantes e indicar un problema serio. Si se utiliza la opción *draft* en la clase de documento (véase el apartado 6.6), los lugares en los que el ancho de línea se ha superado, aunque casi no se perciba visualmente, aparecerán marcados con un cuadro negro en la impresión.

§3.3 Párrafos centrados o alineados por un solo lado



EN un texto normal (es decir, que no sea el título de un capítulo, sección, etc.) L^AT_EX va componiendo líneas justificándolas por ambos lados. Ése es su ambiente o «entorno» por defecto. Si se desea que un texto aparezca centrado hay que introducirlo dentro de un entorno *center*. Los entornos en L^AT_EX son zonas del documento fuente, delimitadas por los comandos *\begin{center}* y *\end{center}*, que sirven para transmitir instrucciones específicas al compilador L^AT_EX sobre su comportamiento en el interior de tales entornos, que puede ser distinto del comportamiento «habitual».

```
\begin{center}
Texto
\end{center}
```

El entorno *center* indica a L^AT_EX que el *Texto* en el interior de dicho entorno no debe estar justificado por ambos lados utilizando división silábica, si fuera necesario, sino que debe estar centrado. Si el *Texto* fuera muy largo, L^AT_EX se ocuparía de partirla y crear varias líneas. Pero podría ocurrir que la forma de partir las líneas que él adopta no fuera la más adecuada para un texto concreto; en ese caso hay que indicarle el lugar por el que debe saltar de línea para centrar

las diferentes líneas. Para ello se utiliza el comando `\\\` tantas veces como se desee. Se puede utilizar el parámetro opcional del comando `\\\` para que, además de cambiar de línea y seguir centrando, la nueva línea esté separada de la anterior un cierto espacio. No hay inconveniente en que *Texto* contenga más de un párrafo.

EJEMPLO 3.2

```
\begin{center}
El ingenioso hidalgo\\
D. Quijote de la Mancha\\[.3cm]
Miguel de Cervantes Saavedra
\end{center}
```

Centrar horizontalmente

El ingenioso hidalgo
D. Quijote de la Mancha

Miguel de Cervantes Saavedra

El entorno `center` puede utilizarse también para centrar otro tipo de objetos como gráficos, tablas, etc., de los cuales nos iremos ocupando en posteriores lecciones. Hasta ahora nos hemos ocupado de centrar horizontalmente; pero también veremos en la lección 18 (apartado 18.5) cómo centrar objetos en sentido vertical.

`LATEX` deja un espacio adicional por encima y por debajo de los textos incluidos en este entorno. En ocasiones eso puede estar desaconsejado. Si se desea centrar un texto en una línea que se inicia, sin dejar espacio adicional alguno, se puede emplear el comando

`\centerline{Texto}`

Si se utiliza este comando, hay que cerciorarse de que el *Texto* no supere la anchura disponible para la línea.

Una estructura parecida a la de centrar un texto es la de llevar un texto al margen de la derecha o de la izquierda (texto en bandera por la izquierda y derecha, respectivamente). Los entornos correspondientes son:

```
\begin{flushright}
Texto
\end{flushright}
```

```
\begin{flushleft}
Texto
\end{flushleft}
```

EJEMPLO 3.3

```
\begin{flushleft}
Probando \\ la forma de alinear\\
por la izquierda
\end{flushleft}
\begin{flushright}
Y ahora \\ de alinear \\ por la derecha
\end{flushright}
```

Textos en bandera

Probando
la forma de alinear
por la izquierda

Y ahora
de alinear
por la derecha

Análogos al comando `\centerline` existen los comandos

`\rightline{Texto}`

`\leftline{Texto}`

cuyos nombres son autoexplicativos, ¡para los que sepan inglés!

Los entornos `flushleft`, `center` y `flushright` están construidos, respectivamente, con los siguientes comandos primitivos de `TeX`:

<code>\raggedright</code>	<code>\centering</code>	<code>\raggedleft</code>
---------------------------	-------------------------	--------------------------

Estos comandos pueden ser de utilidad para construir textos con alineación específica dentro de determinadas estructuras que así lo aconsejen. En el ejercicio 12.2 de la lección 12 podrá encontrar una aplicación del comando `\raggedright`.

§3.4 Párrafos especiales: citas textuales y poemas

La estructura en columnas (apartado 12.3), las tablas (lección 14) y los entornos `minipage`, para construir pequeñas páginas dentro de un párrafo (descritos en la lección 18), permiten construir párrafos más complejos que los entornos `flushleft`, `center` y `flushright` hasta ahora considerados. En este apartado nos ocupamos de un par de tipos especiales de párrafo, de notable utilidad: citas textuales y versos.

Para resaltar una cita textual larga incluida en un documento suelen utilizarse procedimientos diferentes (que a veces aparecen combinados): cambiar el perfil de la letra o el tamaño, o bien modificar la longitud de línea para el texto citado en relación con la longitud de línea en el texto principal. En esta última forma de proceder, la enfatización se consigue mediante un párrafo con formato especial. `LaTeX` implementa para ese propósito los entornos `quotation` y `quote`. Ninguno de ellos modifica el tipo y el tamaño de letra y las diferencias entre ambos provienen de los distintos valores asignados a ciertas longitudes: `quotation` se limita a modificar los márgenes izquierdo y derecho del texto de la cita, mientras que `quote` suprime además la sangría de los párrafos e incrementa la separación entre éstos.

<code>\begin{quotation}</code>	<code>\begin{quote}</code>
<i>Texto</i>	<i>Texto</i>
<code>\end{quotation}</code>	<code>\end{quote}</code>

El resultado que producen estos entornos hace innecesario —así lo señala la Real Academia Española— entrecomillar las citas; algunos autores acostumbran a disminuir ligeramente el cuerpo de la letra en tales situaciones. Tal actuación habrá de realizarse, en su caso, en el interior del entorno mediante los comandos descritos en la lección 5.

EJEMPLO 3.4

```
... Bertolt Brecht expresó su
pensamiento de este modo:
\begin{quotation}
Hay personas que luchan un día, y son bu-
enas. Hay otras que luchan un año y son mejo-
res. Hay quienes luchan muchos años, y son
muy buenas.
\par ... la vida:
éas son las imprescindibles.
\end{quotation}
En esta velada queremos presentar ...
```

... Bertolt Brecht expresó su pensamiento de este modo:

Hay personas que luchan un día, y son bue-
nas. Hay otras que luchan un año y son mejo-
res. Hay quienes luchan muchos años, y son
muy buenas.

Pero hay algunas que luchan toda la vida:
éas son las imprescindibles.

En esta velada queremos presentar a una serie de personas que
encarnan con su vida la formulación que Bertolt Brecht...

El entorno `quotation` es el aconsejable para la inclusión de citas textuales largas

EJEMPLO 3.5

```
... el escritor Bertolt Brecht expresó su
pensamiento de este modo:
\begin{quote}
Hay personas que luchan un día, y son
buenas. Hay otras que luchan un año y
son mejores ... \par ... la vida:
éas son las imprescindibles.
\end{quote}
En esta velada queremos presentar ...
```

... Bertolt Brecht expresó su pensamiento de este modo:

Hay personas que luchan un día, y son muy
buenas. Hay otras que luchan un año y son me-
jores. Hay quienes luchan muchos años, y son
muy buenas.

Pero hay algunas que luchan toda la vida: éas
son las imprescindibles.

En esta velada queremos presentar a una serie de personas que
encarnan con su vida la formulación que Bertolt Brecht...

Para citas textuales breves es preferible el entorno quote

La escritura de versos también requiere la utilización de párrafos especiales. El entorno `verse` se adapta a tal fin. Cada nuevo verso se separa del anterior mediante `\\"`, y las estrofas se separan entre sí como si se tratara de párrafos. Como puede observarse en el poema de G. Celaya del ejemplo 3.6, si un verso no cabe en una única línea L^AT_EX lo parte en dos líneas e inserta una sangría adicional en la segunda de tales líneas, sin que sea necesario realizar ninguna actuación al escribir el texto fuente.

```
\begin{verse}
Verso1 \\
Verso2 \\
...
\end{verse}
```

El entorno resulta bastante confortable en general, pero algunos usuarios echarán en falta la posibilidad de controlar de forma sencilla la separación entre las diferentes estrofas (párrafos): L^AT_EX fija internamente dicha separación, ignorando los eventuales intentos que el usuario pueda realizar para modificar la longitud `\parskip`. Este funcionamiento tan rígido puede resultar asfixiante para la libertad del poeta, que tal vez prefiera utilizar el truco que consiste en hacer uso del argumento optativo del comando `\\"`, en lugar del `\par`, para incrementar la separación entre diferentes estrofas, eludiendo así el encorsetamiento de este entorno. Véase, no obstante, a este respecto el apartado PARA SABER MÁS de esta lección.

EJEMPLO 3.6

```
\dots como botón de muestra de este tipo...
\begin{verse}
Maldigo la poesía concebida como un lujo\\
cultural por los neutrales \\
que, lavándose las manos,
se desentienden y evaden.\\
Maldigo la poesía de quien no
toma partido hasta mancharse \\
\rightline{(G. Celaya)}
\end{verse}
... realizado para esta antología.
```

...como botón de muestra de este tipo de poesía pueden servir estos versos:

Maldigo la poesía concebida como un lujo
cultural por los neutrales
que, lavándose las manos, se desentienden y
evaden.

Maldigo la poesía de quien no toma partido
hasta mancharse

(G. Celaya)

En este poema Celaya refleja muy bien el espíritu que alienta la selección que hemos realizado para esta antología.

Inclusión de versos

§3.5 Interlínea

AS antiguas máquinas de escribir utilizaban un único tipo de letra y cada nueva línea estaba separada de la anterior un cierto espacio. La separación entre líneas podía ser incrementada en factores 0,5 de sí misma, utilizándose la siguiente terminología, que no ha caído en desuso: texto a «un espacio», a «espacio y medio» a «doble espacio»... Con L^AT_EX las posibilidades son mayores. Por defecto L^AT_EX selecciona la interlínea a «un espacio», pero es posible cambiarla en el preámbulo mediante el siguiente código:

```
\renewcommand*\baselinestretch{Número}
```

El valor por defecto de *Número* es 1 y puede ponerse cualquier valor decimal, como, por ejemplo, 1.25. El nuevo interlineado afecta, en principio, a todo el documento.

Por si alguna vez necesitara ir cambiando el valor de la interlínea en diferentes partes de un mismo documento (nosotros nunca hemos sentido esa necesidad) le conviene saber que el paquete *doublespace*, de Stephen Page [52], proporciona un método sencillo para hacerlo. A tal fin necesita cargar dicho paquete en el preámbulo del documento, junto a los demás, mediante

```
\usepackage{doublespace}
```

y a partir de entonces el valor de \baselinestretch será 2. El paquete implementa también el entorno *spacing* que hace posible escribir grupos de texto con un interlineado específico para cada uno de ellos.

```
\begin{spacing}{Número}
Texto
\end{spacing}
```

Ejercicios

- [3.1] Compile un documento que contenga un entorno *quote* como el que se incluye en *ejercicio3.1.tex*. A continuación introduzca el comando \raggedleft dentro de dicho entorno. Compile de nuevo y observe el resultado. Haga lo mismo con el comando \centering.
- [3.2] Los logotipos T_EX y L^AT_EX se consiguen respectivamente con \TeX{} y \LaTeX. ¿Cómo pueden obtenerse «T_EXperto» y «T_EX experto»? ¡Son especies en expansión!
- [3.3] En un texto de varias líneas cambie la interlínea a doble espacio, espacio y medio, 1.27 espacios. ¿Qué ocurre si se pone un valor de interlínea inferior a 1?
- [3.4] Hemos visto en esta lección algunos párrafos especiales, pero pueden construirse más; en la lección 18 veremos otros ejemplos relacionados con las cajas. Puede encontrar un párrafo muy especial, con forma de corazón, en el archivo *ejercicio3.4.pdf*; construirlo es muy sencillo utilizando un paquete adecuado (el paquete *shapepar*). El archivo *ejercicio3.4.tex* contiene el código fuente.

PARA SABER MÁS

- ▶ Hay algunas situaciones en las que L^AT_EX no sangra la primera línea de un párrafo; consulte este aspecto en la lección 11 (entornos `enumerate` e `itemize`) y en la lección 18 (entorno `minipage` y comando `\parbox`).
- ▶ El cuerpo del entorno `verse` está sangrado con relación al texto ordinario y, además, aparece un espacio de separación antes y después del mismo. Esto puede, en ocasiones, ser un inconveniente debido a la pérdida de espacio que supone. Para modificar este comportamiento véase la sección II.3.2.2.
- ▶ Respecto al problema de separación de estrofas en el entorno `verse` comentado en la lección, el truco ofrecido en la página 28 para resolverlo es una solución de compromiso en este estado de iniciación a L^AT_EX; existe una vía sistemática, y de más amplio campo de influencia, para alterar el comportamiento problemático, así como, por ejemplo, para sangrar la primera línea de cada estrofa; véase también la sección II.3.2.2 sobre parámetros de control de las listas.

Caracteres reservados y signos ortográficos



OBJETIVOS

- Conocer los diez caracteres reservados por L^AT_EX para su sintaxis.
- Introducir algunos de los signos ortográficos usuales en español.

PAQUETES COMENTADOS: eurosym

ESCRIBIR texto normal es fácil, con casi todos los editores. De vez en cuando, escribiendo y escribiendo, nos encontramos ante una dificultad: un signo raro, un carácter que no localizamos en el teclado, por ejemplo, o, mucho menos frecuentemente, un signo típico de idiomas para nosotros exóticos. Estos «signos raros», aunque quizás no tanto, serán tema principal en esta lección. En L^AT_EX algunos de estos signos tienen un significado especial, son parte del lenguaje, de la sintaxis; fueron elegidos como parte del lenguaje precisamente por ese carácter de rareza, de forma que no hubiera frecuente confusión entre signos ortográficos y signos del lenguaje de L^AT_EX.

§4.1 Los diez caracteres reservados

HASTA aquí hemos introducido en nuestro computador algunos comandos de L^AT_EX: hemos podido verificar que todos se inician con el carácter \; se trata del carácter reservado más importante para L^AT_EX. También hemos utilizado código para, por ejemplo, centrar: \begin{center}; las llaves (\{}) son importantes en esa sintaxis, son también caracteres reservados que normalmente delimitan un objeto de interés: el valor de un argumento, un bloque (o grupo) de algún tipo dado, etc. ¿Por qué decimos que son «caracteres reservados»? Sencillamente porque tienen un significado especial en el lenguaje: cada uno de ellos representa una orden, una tarea a realizar, una interpretación particular... Son elementos propios del lenguaje de L^AT_EX.

En total se contabilizan diez caracteres reservados para L^AT_EX, a saber:

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

De los tres primeros ya sabemos un poco. En cuanto a los restantes nos limitaremos, en este punto, a una breve descripción de su uso.

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

Cuadro 4.1: Comandos para introducir en el texto los signos correspondientes a los diez caracteres reservados. En el ejemplo 4.1 descubrirá un detalle adicional de \%

- [#] Se utiliza para nombrar a los argumentos de un comando (véase la página 94).
- [&] Se utiliza para separar columnas de una tabla (véase la página 111).
- [%] Se utiliza para incluir comentarios en el fichero fuente. En esta misma lección (apartado 4.2) se describe con mayor precisión.
- [~] Se utiliza para evitar la separación de palabras; para más detalles véase el apartado 4.3.
- [\$ _ ^] Se utilizan en las fórmulas matemáticas (véase la lección 16).

Observemos que todos estos caracteres reservados aparecen con poca frecuencia en un texto ordinario; quizás los más frecuentes son el signo porcentual y las llaves, éstas en fórmulas matemáticas. El más raro es sin duda la «antibarra» (\); por ello Donald Knuth lo eligió como el carácter que inicia un comando. Aunque aparezcan raramente, la cuestión es: cuando los necesitemos ¿cómo podremos conseguir escribirlos en un texto? La respuesta se encuentra en el cuadro 4.1, en el que se muestra cómo cada signo puede ser introducido mediante un comando. Observe, en dicho cuadro, que los comandos que escriben los signos ~ y ^ van seguidos de llaves: la explicación de esta simetría es que los comandos \~{} y \^{} están previstos para actuar como acentos sobre el carácter que les siga; las llaves consiguen ubicar los signos sin necesidad de que siga un carácter. De hecho existe una alternativa para cada uno de estos dos caracteres, que viene dada por los comandos:

<code>\textasciitilde</code>	<code>\textasciicircum</code>
------------------------------	-------------------------------

Observe que los resultados no son exactamente iguales: \textasciitilde produce ~, mientras que \~{} produce \~{}; en el segundo caso, ^ y ^ son los resultados de \textasciicircum y \^{}, respectivamente¹.

§4.2 Comentarios en el código fuente

COMO hemos aprendido en el apartado anterior, el signo porcentual (%) es un carácter reservado. El efecto que causa es el siguiente: todo lo que en el documento fuente sigue al %, hasta el final de la línea de código en la que aparece, es ignorado al componer el documento; todo eso se toma como inexistente.

¹Esta diferencia no se da en caso de optar por la opción OT1 del paquete fontenc, en lugar de la opción T1 (véase la lección 2).

Este signo se puede utilizar, por tanto, para incluir comentarios en el fichero fuente. Estos comentarios pueden ser de interés por distintas causas: pueden servir para aclarar el efecto de comandos que no conocemos bien, para introducir espacios que separen visualmente distintas partes del código (sin que se conviertan en blancos del texto), o simplemente para comentarios que el propio autor realiza sobre su escrito. La utilización de estos comentarios puede ser de gran utilidad, especialmente para los que se inician con L^AT_EX.

Uno de los usos más interesantes del % es el siguiente, ilustrado en el ejemplo 4.1. Ya conocemos que L^AT_EX interpreta un final de línea en el código fuente como un espacio usual. En ocasiones (muy a menudo en la definición de comandos, véase la lección 10) es conveniente romper la línea en el código fuente, sin que esto suponga un espacio (por ejemplo, para hacer más legible el código); para conseguirlo basta terminar la línea con un carácter %, sin espacios previos. Esta estrategia aparece a menudo en los ejemplos de este libro.

EJEMPLO 4.1

El carácter \negthinspace\% se utiliza para introducir comentarios que no son compilados.

%

Observe que la línea de código anterior no ha iniciado nuevo párrafo, es ignorada completamente ya que comienza con un \negthinspace\%.

Ahora podemos cortar una palabra, sin introducir espacios.

Compare los resultados de 10\% y 10\negthinspace\%.

El comando...

El carácter % se utiliza para introducir comentarios que no son compilados. Observe que la línea de código anterior no ha iniciado nuevo párrafo, es ignorada completamente ya que comienza con un %.

Ahora podemos cortar una palabra, sin introducir espacios.

Compare los resultados de 10 % y 10%.

El comando \% introduce un pequeño espacio antes del %; mientras que el comando \negthinspace anula dicho espacio. Este pequeño espacio es introducido por la opción spanish de babel, para seguir las normas tipográficas españolas. Lo natural es anular dicho espacio si un número no precede al %.

Comentarios en el código fuente con %. El comando \negthinspace es explicado en el apartado 7.3

§4.3 Evitar la separación de palabras

SABEMOS ya que el compilador T_EX siempre justifica las líneas (a menos que le indiquemos lo contrario) y lo hace a su manera, a través de un complicado algoritmo. Esto significa que el que escribe no posee control alguno (al menos en estos niveles actuales de conocimiento elemental) sobre los lugares en los que un texto será dividido en líneas para formar un párrafo. En ocasiones, por el contrario, es deseable que dos palabras no sean separadas, es decir, que una línea no acabe justamente entre ellas. Es una conocida norma tipográfica no separar términos entre los que existe una dependencia o que son complementarios, como por ejemplo: Sr. Director, capítulo 3, o A. Einstein².

²Para otras situaciones en que se recomienda no separar palabras, véase el término *separación de palabras* en [60].

Para conseguir que L^AT_EX no separe estos términos basta unirlos con el carácter reservado ~, denominado *virgulilla* o circunflejo griego³. Así, siguiendo los ejemplos anteriores, podríamos evitar su separación introduciendo el texto fuente: Sr.~Director, capítulo~3, o A.~Einstein. Pero además esto puede ser concatenado para ligar más palabras, por ejemplo: Donald~E.~Knuth.

También es de interés evitar la separación de palabras para eliminar las llamadas «palabras viudas», que son palabras que aparecen como la única palabra de la última línea de un párrafo. Si estas palabras viudas son muy cortas (a veces se dice que con cinco o menos letras), la línea se considera tipográficamente incorrecta, véase [61]. Si una palabra viuda se une a la anterior con el carácter ~, entonces dejará de ser viuda; además, el párrafo no sufrirá grandes alteraciones de composición ya que, salvo quizás en caso de que posea sólo dos líneas, el espacio perdido al bajar una palabra a la última línea podrá ser repartido por L^AT_EX sin generar grandes blancos. Realmente este procedimiento de soldar las dos últimas palabras del párrafo no siempre evita que la última línea sea corta; de hecho, en ocasiones, produce una línea aún más corta, pues el compilador divide silábicamente la última palabra, dejando sólo parte de ella en la última línea.

§4.4 Signos ortográficos

El origen de T_EX y L^AT_EX es anglosajón; como consecuencia, no se contemplaron en ellos, como parte esencial, signos ortográficos de lenguas distintas al inglés. Por ello, en su inicio, planteaban inconvenientes con las letras acentuadas o con los signos de entonación iniciales (¡ ¢), todos ellos usuales en español. Las posibilidades actuales para incluir todos estos signos son variadas y por el momento no van a ser abordadas todas ellas (véase el apartado PARA SABER MÁS de esta lección). Por ahora nos contentaremos con aprender la forma más sencilla de introducirlos: directamente desde el teclado, es decir, sin escribir comandos. Algunos de ellos, sin embargo, no están disponibles en el teclado habitualmente, por lo que mostraremos los comandos que permiten introducirlos.

En efecto, la mayor parte de los signos ortográficos usuales en español pueden introducirse en el documento fuente desde el teclado, *siempre y cuando* estemos utilizando el paquete inputenc con la opción adecuada al editor utilizado. Esta condición merece una explicación más detallada. Hemos convenido en la lección 2 que el preámbulo estándar para todas estas lecciones incluya:

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

Suponemos con ello que el lector va a escribir su documento fuente en un sistema operativo que maneja la página de códigos latin1 (como MS-WINDOWS⁴ o LINUX) y con un editor arbitrario que, también, es capaz de leer y escribir ficheros con dicha codificación. Si no es éste el caso, la opción latin1 deberá ser sustituida por una adecuada al entorno de trabajo utilizado (véase el

³El carácter ~ no suele estar presente en los teclados. Una forma cómoda de escribirlo es recurriendo a su código ASCII, que es el 126; para ello, bajo MS-WINDOWS, basta, manteniendo pulsada la tecla ALT, teclear 126 en el teclado numérico. Una alternativa es teclear ALT GR 4 seguido de la barra espaciadora; mientras que bajo LINUX basta pulsar ALT GR 4 o ALT GR ñ.

⁴La página de códigos de MS-WINDOWS no es latin1, pero coincide en una buena parte de los caracteres con ella.

apartado PARA SABER MÁS del capítulo 1). Pues bien, si se dan las condiciones anteriores, la mayor parte de los signos ortográficos introducidos desde el teclado se escribirán en el documento correctamente. Así, por ejemplo, desde un teclado estándar para PC con el sistema LINUX y desde un editor que utiliza la página de códigos latin1, pueden introducirse desde el teclado todos los caracteres acentuados (con acentos agudo, grave o circunflejo), así como los signos que figuran a continuación, donde se muestra, en la línea superior, el signo introducido y, en la inferior, el resultado que produce⁵:

º	ª		"	@	.	í	'	¿	[]	ç	ç	<	>
º	ª		"	@	.	í	'	¿	[]	ç	ç	<	>

En la lista anterior algunos signos son compartidos por todos los sistemas y editores⁶ (como " []) y, al contrario, no se han incluido otros signos usuales, con idéntico carácter universal, como, por ejemplo: paréntesis, el signo de igualdad, exclamación e interrogación de cierre, etc. Tampoco se han incluido signos que corresponden a alguno de los diez caracteres reservados, ya que su introducción directa provocaría errores de compilación (y ya sabemos cómo escribirlos).

Con todo ello, ¿quedó resuelto cualquier problema que se refiera a la introducción de signos ortográficos? La respuesta es claramente negativa, por distintas razones. En primer lugar, algunos signos ortográficos usuales no están disponibles en el teclado (por ejemplo «»). En segundo lugar, algunos signos disponibles en el teclado causan problemas de compilación si se introducen directamente; el ejemplo más llamativo es el signo del euro (€). En tercer y último lugar, algunos signos o tienen varias formas, no equivalentes, de ser introducidos o poseen variantes que responden ya sea a alguna característica de LATEX o a alguna tradición tipográfica (por ejemplo, las comillas). De todos estos problemas vamos a ocuparnos a continuación.

Comillas

En general en los textos se manejan dos tipos de comillas: las latinas, a veces denominadas también comillas españolas o francesas, dobles («») o simples (<>), y las inglesas, dobles (" ") o simples (' '). Mucho menos usuales en español, aunque habituales en alemán, son las «comillas bajas» (como en „ß”).

La forma de introducir todas estas comillas directamente desde el teclado se muestra en el cuadro 4.2 (recuerde que la posibilidad de uso del teclado depende del paquete inputenc y que el resultado puede depender de la opción T1 del paquete fontenc). Observe que siempre son diferentes las comillas de abrir y de cerrar (el uso de unas comillas únicas, las mismas para abrir y cerrar, es uno de los errores característicos de los novatos con LATEX).

⁵El resultado que muestra esta lista de signos depende también de la opción T1 del paquete fontenc, opción que suponemos utilizada.

⁶Éstos son los caracteres cuyo código se encuentra entre los 128 primeros de todas las páginas de códigos.

Teclado	Resultado
<<	« ^a
>>	»
``	“
”	”
~	‘
,	’

Cuadro 4.2: Comillas y cómo obtenerlas directamente desde el teclado. Es conveniente una aclaración: el signo ~ representa el acento grave del teclado, mientras que ' representa el apóstrofo. Observe que no se utiliza en ningún momento el carácter ", disponible en el teclado, pero cuyo uso no es recomendable

^aEn LINUX se pueden incluir « y » mediante las combinaciones de teclas ALT GR z y ALT GR x, respectivamente.

Teclado	Resultado
-	-
--	—
---	—

Cuadro 4.3: Guiones introducidos directamente desde el teclado

La tradición tipográfica española asigna una prioridad en el uso de los tipos de comillas: siempre las latinas (dobles); si se necesita entrecomillar en un texto que ya está entrecomillado, para el texto interior se utilizan las inglesas dobles y, en un posible tercer nivel, las inglesas simples.

Guiones

\LaTeX distingue tres tipos de guiones: corto (-), medio (–) y largo (—). Al margen de estos guiones, empleados en el texto ordinario, se encuentra el signo de la substracción o resta (–), que es distinto de los tres anteriores y se reserva para las expresiones matemáticas (véase la lección 16).

En tipografía española sólo se suelen considerar el corto (denominado guión, a secas) y el largo (denominado raya). El primero se utiliza para palabras compuestas y división de palabras por sílabas. La raya posee varios usos: incisos en el texto, diálogos, sustitución del término inmediatamente anterior, que se repite, en catálogos, tablas y bibliografía, etc. Entre los usuarios de \LaTeX el guión medio suele ser utilizado para indicar intervalos numéricos, como en ‘23–33’. Si conviene utilizarlo o no en español es cuestión de cada uno.

La introducción de estos guiones no puede ser más sencilla, viene indicada en el cuadro 4.3.

Puntos suspensivos

En español los puntos suspensivos suelen ser tipografiados como tres puntos seguidos, sin espacio adicional alguno entre ellos. \LaTeX concibe la posibilidad de aumentar el espacio que sigue a un punto, respecto al espacio normal entre palabras, posibilidad adaptada a las normas

tipográficas anglosajonas. De no ser por esta circunstancia especial no tendría sentido dedicar este apartado a los puntos suspensivos.

Disponemos de tres formas de introducir los puntos suspensivos, a saber:

...

\...

\dots

La primera y más sencilla (tres puntos, desde el teclado) funciona siempre, pero puede dar un espacio incorrecto en español después de los tres puntos (esto no ocurre con nuestro preámbulo estandarizado, la opción `spanish` de `babel` se encarga de ello). El comando `\...` siempre proporciona el espacio correcto, pero sólo funciona si está cargado `babel` con opción `spanish` (en caso contrario ocurrirá un error en la compilación). El comando `\dots` es el propio de `LATEX`, siempre está disponible, pero el espacio entre los puntos no es el que recomiendan las normas tipográficas españolas.

EJEMPLO 4.2

Paquetes, paquetes,... y más paquetes\\
 Paquetes, paquetes,\... y más paquetes\\\\
 Paquetes, paquetes,\...y más paquetes\\\\
 Packages, packages,\dots and more packages

Paquetes, paquetes,... y más paquetes
 Paquetes, paquetes,... y más paquetes
 Paquetes, paquetes,...y más paquetes
 Packages, packages,... and more packages

Puntos suspensivos. Observe que `\...` no necesita un espacio posterior y que, a la vez, el espacio que le sigue es respetado

Ordinales y grados

Ya hemos aprendido que los ordinales abreviados, como por ejemplo 1^{a} o 1^{o} , pueden ser introducidos desde el teclado (basta teclear 1^{a} o 1^{o}). ¿Qué hacer para obtener 1^{er} ? `LATEX` nos ofrece para ello el comando:

`Voladitas`

El nombre, aparentemente complicado, de este comando merece quizás una breve explicación. El término *superscript* se refiere, en terminología `LATEX`, a exponente; como los exponentes son típicos de fórmulas matemáticas, no de texto ordinario, se antepone el prefijo *text*. Con este comando la introducción de ordinales o, más generalmente, de letras *voladitas* es muy simple: el código `2A` proporciona la salida 2^{A} , mientras que `3er` es el resultado de `3er`.

Con `babel`, opción `spanish`, disponemos de un comando alternativo para estas letras voladitas:

`\sptext{Voladitas}`

La diferencia entre ambos comandos es doble: la versión específica para español (`\sptext`) introduce un punto antes de las voladitas (una norma tipográfica española) y cambia el tamaño de las mismas, si éstas son mayúsculas. Compare los resultados: el código `2\sptext{A}` proporciona la salida $2.^{\text{A}}$, mientras que `3.er` es el resultado de `3\sptext{er}`.

Una excepción a la grafía de las letras voladitas la constituye el signo utilizado para indicar los grados (angulares o de temperatura); para ello disponemos del comando

Comando	Resultado	Comando	Resultado
\oe	œ	\OE	Œ
\ae	æ	\AE	Æ
\dag	†	\ddag	‡
\S	§	\P	¶
\textbullet	•	\textvisiblespace	␣
\textregistered	®	\copyright	©
\texttrademark	™	\pounds	£

Cuadro 4.4: De todos estos signos el menos usual es, sin duda, el generado por \textvisiblespace (␣). Ya apareció en la lección anterior y representa un espacio. Su principal, y casi único, uso es el de representar la necesidad de un espacio en blanco, algo usual en lenguajes formales como, por ejemplo, lenguajes de programación

\textdegree

Observe la diferencia entre los grados y el ordinal masculino: el código 3\textdegree genera 3°, mientras que 3o produce el resultado 3º.

Otros signos

En el cuadro 4.4 se muestran los comandos adecuados para generar algunos signos usuales en la escritura española. Para la inclusión de otros signos más típicos de idiomas no latinos, véase el apartado PARA SABER MÁS de esta lección.

Algunos de estos signos pueden introducirse directamente desde el teclado (aunque esta posibilidad puede depender de las opciones elegidas para inputenc y fontenc). El problema es que no están disponibles en el teclado, pero pueden obtenerse si su editor posee una herramienta, normalmente un comando de menú, que le permite incluir todos los caracteres de la página de códigos utilizada, o bien si conoce su código en dicha página de códigos. Así, por ejemplo, el calderón (¶) posee el código 182 en MS-WINDOWS, y puede obtenerse tecleando ALT 0182 en el teclado numérico. En cualquier caso, no incluiremos aquí la lista de los signos alcanzables de este modo.

Los comandos \copyright y \textregistered son ejemplos de uso de un comando con un ámbito de aplicación más amplio:

\textcircled{Carácter}

Conviene que el argumento *Carácter* sea un único carácter, que será rodeado por una circunferencia. Así \textcircled{a} produce ®. El resultado no es bueno si *Carácter* es una consonante con mayor altura o profundidad que las vocales, por ejemplo en ® o ® (la letra ‘p’ posee una «profundidad» no nula, va más abajo de la línea imaginaria sobre la que descansan todas las letras, véase el apartado 18.1). Tampoco es bueno el resultado si *Carácter* es una mayúscula. Estos problemas pueden evitarse reduciendo el tamaño de los tipos, en el primer caso, o utilizando versalitas en el segundo (véase la lección siguiente).

Euros

Resulta bastante sorprendente que, avanzado el año 2003, L^AT_EX no contemple el acceso inmediato y transparente al símbolo del euro. Ignoramos las razones que motivan este escandaloso retraso, pero la solución al problema que plantea no es muy complicada.

Para poder introducir el símbolo del euro en nuestros documentos de L^AT_EX hemos de realizar dos operaciones. La primera, cargar el paquete *eurosym*. Esta tarea es sencilla: basta incluir la siguiente línea

```
\usepackage{eurosym}
```

en el preámbulo de nuestro documento (por cuestiones de orden en el código fuente, lo mejor es incluirla junto a otras líneas del tipo *\usepackage*).

Una vez incluido este paquete basta utilizar, donde lo deseemos, el comando

```
\euro
```

que inserta el símbolo oficial del euro (€).

En los teclados estándar el símbolo del euro suele estar disponible, pero con las acciones anteriores no podremos incluirlo directamente desde el teclado. Si nos empeñamos en conseguir esta posibilidad, podemos hacerlo trabajando un poco más. Para ello bastaría incluir en el preámbulo del documento la línea:

```
\DeclareInputText{128}{\euro}
```

si estamos trabajando en MS-WINDOWS, y la línea

```
\DeclareInputText{164}{\euro}
```

si nuestro entorno de trabajo es LINUX. La explicación de este código es sencilla. El símbolo del euro se identifica como el carácter 128 en la página de códigos de MS-WINDOWS y como 164 en la de LINUX. Con la línea correspondiente le pasamos a L^AT_EX la orden de que el carácter con dicho código sea identificado con el comando *\euro*. A decir verdad, este mismo procedimiento se podría utilizar con otros signos (como, por ejemplo, el calderón, ¶), en la línea de ideas comentada en el apartado anterior.

Ejercicios

[4.1]  Cambie la opción T1 de *fontenc* por la opción OT1 en nuestro preámbulo estándar.

Tras el cambio escriba, desde el teclado, los signos de la página 35. ¿Observa diferencias?

¡No olvide volver a la opción T1! En el archivo *Ejercicio4.1.tex* encontrará estos signos, así como algunos comandos *\DeclareInputText*; practique la introducción de otros símbolos desde el teclado ya sea directamente o mediante la inclusión de nuevos comandos *\DeclareInputText*.

[4.2] Escriba el código que figura en la línea siguiente. ¿Qué ocurre?

---- hola -- -- en lugar de --- hola ---

[4.3] Mediante el uso del comando *\textcircled* intente obtener el signo ®. ¡Necesitará el comando *\textsc* de la lección siguiente!

4.4 ¡No lo deje para después!... Consiga escribir el símbolo oficial del euro en un documento. Probablemente necesitará instalar el paquete `eurosym`; consulte el apartado PARA SABER MÁS de la lección 2.

PARA SABER MÁS

- En la página 34 hemos comentado que soldar las dos últimas palabras del párrafo no siempre evita la aparición de palabras viudas. Para aumentar las posibilidades de evitar este fenómeno disponemos de herramientas que permiten mayor control sobre la formación de los párrafos, véase el apartado II.1.1.2.
- Los signos matemáticos forman un capítulo aparte, véanse la lección 16 y el capítulo 5.
- Cuándo, cómo y por qué `LATEX` puede modificar el espacio que sigue a un punto, en II.1.1.1.
- Para saber cómo proceder en caso de querer escribir un documento que utiliza dos o más idiomas (es decir, para aprender todo sobre `babel` y sus opciones, incluidas `basque`, `catalan`, `galician` y `spanish`) véanse las secciones II.1.2 y II.1.3.

Lección 5



OBJETIVOS

- Los tres atributos fundamentales de una fuente: familia, perfil y grosor.
- Principales familias, perfiles y grosores.
- Otras características importantes: tamaño y color.

PAQUETES COMENTADOS: color

CUANDO nos ponemos delante de un editor de textos, por muy sencillo que sea, una de las primeras elecciones que debemos hacer es el tipo de letra con el que vamos a escribir. Las diferentes posibilidades dependen de la potencia del editor, aunque la mayoría nos permite incluso incorporar nuevos tipos. Pero, ¿qué características posee una fuente o tipo? Básicamente son tres las propiedades que caracterizan un tipo de letra: su familia, su perfil y su grosor.

En L^AT_EX también es posible, por supuesto, seleccionar el tipo de letra que más nos guste, y para ello debemos saber cómo indicarlo. Veremos a continuación cuáles son las instrucciones necesarias.

Los tipos que, por defecto, emplea L^AT_EX los creó Donald E. Knuth para su utilización con T_EX y los llamó Computer Modern Fonts. Son diferentes de los que se utilizan en las imprentas, aunque se incluyen tipos con diferentes atributos, entre ellos los necesarios para las fórmulas matemáticas. No obstante, es posible utilizar otros tipos diferentes (como las fuentes PostScript), con lo que se amplía la flexibilidad de L^AT_EX (véase el apartado PARA SABER MÁS).

§5.1 Las diferentes familias

R_F S_F T_F

BÁSICAMENTE, L^AT_EX dispone de tres familias de tipos de letra: roman (normal), sanserif (sin adornos) y tipo máquina de escribir (typewriter). Por defecto utiliza la familia roman en el texto base (como el de este párrafo) y no es necesario indicar explícitamente este tipo de letra. Las otras familias se utilizan, en general, para resaltar un texto en relación con el texto base. Las instrucciones para escribir con una familia concreta son:

\textrm{Texto}	\rmfamily Texto
\textsf{Texto}	\sffamily Texto
\texttt{Texto}	\ttfamily Texto

De las dos columnas anteriores, el modo comando que aparece en la columna de la izquierda se utiliza para textos pequeños (unas palabras o una frase), y nunca más de un párrafo. La sintaxis que se indica en la parte de la derecha es la que conviene utilizar cuando se cambia de familia de letra en un texto con una longitud importante. Es una declaración cuyos efectos se mantienen hasta que se declare una nueva familia o hasta que finalice el «grupo» actual. Un ejemplo del funcionamiento es el siguiente:

EJEMPLO 5.1

Un mismo texto produce un resultado diferente según el tipo de letra.

En sanserif: {\sffamily El mismo texto produce este resultado en sanserif}.\\par
Y en typewriter: {\ttfamily El mismo texto produce este resultado en typewriter}.

Un mismo texto produce un resultado diferente según el tipo de letra.

En sanserif: El mismo texto produce este resultado en sanserif.

Y en typewriter: El mismo texto produce este resultado en typewriter.



Para delimitar un grupo se pueden utilizar las llaves {} (véase el apartado 10.1 para una información más detallada). El contenido del grupo será el único afectado por las acciones que realicen los comandos que sobre él o en su interior se utilicen (obsérvese la utilización de las llaves en el ejemplo 5.1).

Otra forma de conseguir el mismo resultado es con los entornos que cada familia tiene asociados. En concreto, y a modo de ejemplo, esto significa que el resultado que se obtiene con {\sffamily Texto} y el que produce el código que sigue es el mismo:

```
\begin{sffamily}
Texto
\end{sffamily}
```

La familia \ttfamily se utiliza en este libro, en general, para referirse al modo en que está escrito el texto fuente, en particular los comandos.

§5.2 El perfil es importante

S **K_F** **S_F** **K_{A_F}**

CADA una de las familias de tipos anteriores tiene cuatro perfiles diferentes: recto (o normal), *italico*, *inclinado* y VERSALITA (o de letras mayúsculas pequeñas), que corresponden respectivamente a los comandos siguientes:

\textup{Texto}	\upshape Texto
\textit{Texto}	\itshape Texto
\textsl{Texto}	\slshape Texto
\textsc{Texto}	\scshape Texto

La estructura y la utilización de estas dos sintaxis es análoga a la indicada en el apartado anterior para las familias. Por defecto L^AT_EX utiliza el primero de ellos (`\upshape`) como texto base, y no es necesario indicarlo explícitamente a menos que se desee poner unas palabras «rectas» dentro un grupo de texto que esté, por ejemplo, inclinado.

Cuando después de un texto inclinado o itálico viene un texto recto, la parte superior de la última letra del texto inclinado ocupa parte del espacio de separación entre las palabras, con lo que el resultado, estéticamente, no es muy satisfactorio. Para paliar este efecto, al final de un texto inclinado o itálico (incluido con uno de los comandos `\itshape` o `\slshape`) se añade un pequeño espacio adicional (la corrección itálica) que se introduce con el comando `\vphantom{}`, aunque en algunos casos (por ejemplo, si hay una coma o un punto) puede resultar innecesario. Dicha corrección ya se realiza automáticamente cuando se utiliza el modo comando (`\textit` o `\textsl`), que figura en la primera columna.

EJEMPLO 5.2

```
Texto escrito en perfil recto.\\
{\itshape Texto escrito en perfil it\'alico.} \\
{\slshape Texto escrito en perfil inclinado.} \\
{\scshape Texto escrito en perfil versalita.}
```

Texto escrito en perfil recto.
Texto escrito en perfil it\'alico.
Texto escrito en perfil inclinado.
 TEXTO ESCRITO EN PERFIL VERSALITA.

El mismo texto produce resultados distintos en los diferentes perfiles

§5.3 ¿Normal o grueso?

M_s F_s B

FINALMENTE existen dos grosores, también denominados series, para el tipo de letra: normal o medio y grueso o negrita. En realidad, L^AT_EX admite muchos más grosores, aunque por defecto sólo utiliza los dos anteriores. Los comandos para activarlos son:

<code>\textmd{Texto}</code> <code>\textbf{Texto}</code>	<code>\mdseries Texto</code> <code>\bfseries Texto</code>
--	--

Los comentarios relativos a grupos que hemos realizado en el apartado anterior (referidos a las familias) se pueden aplicar aquí también. No obstante, declaraciones como `\sffamily`, `\itshape` o `\bfseries` pueden utilizarse sin estar encerradas dentro de llaves y actúan del mismo modo, salvo que su acción, al no estar restringida a un grupo, se ejerce a partir del momento en que se utiliza y se mantiene hasta que otro comando de la misma clase inicie una actuación diferente.

Los comandos correspondientes a la familia, al perfil y al grosor de la fuente pueden combinarse; por ejemplo, `\bfseries\itshape` produce negrita itálica como ocurre en *este texto*.

EJEMPLO 5.3

```
Este texto es normal. \textbf{Este texto es
negrita \itshape y \'este además it\'alico.} \\
Texto normal \slshape y texto inclinado\/
\upshape que vuelve ahora a normal.
```

Este texto es normal. **Este texto es negrita y \'este además
it\'alico.**

Texto normal y *texto inclinado* que vuelve ahora a normal.

Hay otra modalidad para resaltar o enfatizar un texto dentro de otro:

`\emph{Texto}`

`\em Texto`

Estos comandos actúan del siguiente modo: si el texto ambiente en que se inserta es recto, entonces *Texto* se italiza, y, recíprocamente, si el ambiente es de texto itálico o inclinado, el enfatizado consiste en poner recto el texto, incluyendo la corrección itálica si se utiliza la primera versión.

Existe también la posibilidad de subrayar, pero no se trata de un tipo de letra sino más bien de subrayar un grupo o paquete de objetos, que pueden ser palabras, pero también otras cosas. Su comportamiento es diferente al de una fuente por lo que es poco aconsejable como forma de resaltar un texto. En las composiciones antiguas, donde sólo había un tipo de letra, un trozo de texto subrayado en un manuscrito indicaba al impresor que debía italizarlo.

`\underline{Texto}`

EJEMPLO 5.4

```
Este texto ... ahora \emph{resaltamos} otro...
{\itshape Este ... ahora \emph{resaltamos}...}
{\slshape Este ... ahora \emph{resaltamos}...}
Este es un \underline{ejemplo} de
funcionamiento \underline{del comando}.
```

Este texto es normal y ahora *resaltamos* otro texto.
Este texto está en *italicas* y ahora resaltamos otro texto.
Este texto está *inclinado* y ahora resaltamos otro texto.
Este es un ejemplo del funcionamiento del comando.

Enfatizado y subrayado de un fragmento de texto



§5.4 El tamaño también importa

TAMBIÉN es posible escribir textos o fórmulas en diferentes tamaños utilizando los comandos que se indican a continuación y que se encuentran ordenados, según tamaño decreciente, de arriba abajo y de izquierda a derecha.

`\Huge Texto`
`\huge Texto`
`\LARGE Texto`
`\Large Texto`
`\large Texto`

`\normalsize Texto`

`\small Texto`
`\footnotesize Texto`
`\scriptsize Texto`
`\tiny Texto`

EJEMPLO 5.5

```
esta letra es la normal \par
{\large ésta es grande} \par
{\Large ésta es más grande} \par
{\LARGE y ésta todavía más grande} \par
{\small ésta es pequeña}
{\footnotesize y ésta más pequeña}
```

esta letra es la normal
ésta es grande
ésta es más grande
y ésta todavía más grande
ésta es pequeña y ésta más pequeña

Diferentes tamaños de letra

Los tamaños `\large`, `\Large`, etc., son relativos al tamaño base `\normalsize`. De hecho, el tamaño absoluto de una letra `\normalsize` para una determinada opción de una clase de documento puede corresponder al tamaño `\large` en otra clase o en otra opción de la misma clase (véase el apartado 6.6).

Lo anterior corresponde a la familia por defecto (`\rmfamily`), pero, obviamente, lo mismo puede conseguirse con cualquier otra familia, como se observa en el siguiente ejemplo.

EJEMPLO 5.6

```
{\large\slshape existe un tipo ... grande}\par
{\footnotesize\bfseries y un tipo ... etc.}
```

*existe un tipo inclinada grande
y un tipo negrita más pequeña, etc.*

Combinación de perfiles, grosores y tamaños

Las letras negritas y de tamaños mayores sirven, por ejemplo, para resaltar el título de los capítulos de un libro o las secciones de un artículo. Sólo ocasionalmente se emplean fuera de este ámbito. De esas tareas se ocupa directamente L^AT_EX cuando se le dice que algo es el título de un capítulo o de una sección. Enfatizar un texto dentro de un párrafo sin aumentar el tamaño de las letras (italizándolo, por ejemplo) es una opción de uso más frecuente en la tradición tipográfica.

§5.5 Usando colores

UNO de los efectos más llamativos de los trabajos profesionales de imprenta es, sin duda alguna, el color y, muy especialmente, la impresión de gráficos con calidad fotográfica. La inclusión de gráficos en color (o en blanco y negro) será tratada más adelante, concretamente en la lección 9. Por lo que respecta a la utilización de colores, si queremos colorear nuestro texto debemos utilizar el paquete `color` junto con un «controlador» (frecuentemente llamado «driver»), que se selecciona como una opción del paquete. En definitiva necesitaremos cargar el paquete en la forma:

```
\usepackage[Controlador]{color}
```

La lista completa de los controladores disponibles se incluye en la segunda parte del libro, véase el cuadro II.3.1; por ahora citaremos los dos controladores principales, que son:

`dvips` Utiliza la sintaxis apropiada para que DVIPS genere un archivo PostScript con colores.

`pdftex` Para ser utilizado con el programa PDFL^AT_EX, que genera directamente un archivo PDF.

Por ejemplo, si queremos que nuestra salida sea un documento PDF (para lo cual compilaremos el documento con PDFL^AT_EX), entonces la instrucción correcta sería¹:

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

¹En algunas instalaciones de L^AT_EX no es necesario incluir ninguna opción de controlador, y según el compilador utilizado se selecciona una de las opciones por defecto. Por ejemplo, en las distribuciones de MiK^TE_X y te^TE_X se carga dvips si se compila con L^AT_EX y pdftex si se compila con PDFL^AT_EX.

GreenYellow	Yellow	Goldenrod	Dandelion	Apricot
Peach	Melon	YellowOrange	Orange	BurntOrange
Bittersweet	RedOrange	Mahogany	Maroon	BrickRed
Red	OrangeRed	RubineRed	WildStrawberry	Salmon
CarnationPink	Magenta	VioletRed	Rhodamine	Mulberry
RedViolet	Fuchsia	Lavender	Thistle	Orchid
DarkOrchid	Purple	Plum	Violet	RoyalPurple
BlueViolet	Periwinkle	CadetBlue	CornflowerBlue	MidnightBlue
NavyBlue	RoyalBlue	Blue	Cerulean	Cyan
ProcessBlue	SkyBlue	Turquoise	TealBlue	Aquamarine
BlueGreen	Emerald	JungleGreen	SeaGreen	Green
ForestGreen	PineGreen	LimeGreen	YellowGreen	SpringGreen
OliveGreen	RawSienna	Sepia	Brown	Tan
Gray	Black	White		

Cuadro 5.1:  Nombre de los 68 colores predefinidos con la opción `dvipsnames`. En el CD-ROM que acompaña a este libro puede encontrar el archivo `68colores.pdf` que muestra todos estos colores

Las instrucciones anteriores nos permiten escribir en color, pero sólo con unos pocos colores: rojo (`red`), verde (`green`), azul (`blue`), cian (`cyan`), magenta y amarillo (`yellow`). Si queremos utilizar más colores debemos incluir la opción `usenames` cuando cargamos el paquete:

```
\usepackage [dvips,usenames] {color}
```

De esta manera podremos utilizar los 68 colores originalmente sugeridos por James Hafner, cuyos nombres se recogen en el cuadro 5.1. Éstos son reconocidos tanto por DVIPS como por PDFLATEX. Sin embargo, si queremos utilizarlos con PDFLATEX², tendremos que incorporar además la nueva opción `dvipsnames`:

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

Los comandos para utilizar el color siguen una sintaxis semejante a la de los existentes para el uso de los diferentes tamaños y perfiles de las fuentes, que hemos descrito en los apartados anteriores. Los dos comandos básicos para escribir en colores son los siguientes:

`\color{NombreColor}`

Cambia el color actual al color `NombreColor` hasta que acabe el grupo o entorno en el que estamos, o hasta que vuelva a aparecer otro comando `\color` que lo anule. Es un comando declarativo, del mismo tipo que `\bfseries` o `\Large`.

`\textcolor{NombreColor}{Texto}`

Sirve para imprimir el `Texto` en el color `NombreColor`. Este comando es equivalente, en sus resultados, a `\color{NombreColor}Texto` y es un comando del mismo tipo que `\textbf` o `\textsf`.

²En la versión actual del paquete `color` (v1.0i), para evitar problemas con la opción `pdftex` (v0.03k) es aconsejable comenzar el documento escribiendo `\color{Black}`.

Ejercicios

- [5.1] Escriba un documento, con una extensión de varias líneas, e intente subrayar las dos primeras líneas. ¿Qué ocurre?
- [5.2] Escriba un documento con varios párrafos, introduzca al principio y al final de uno de ellos, respectivamente, los comandos `\begin{large}` y `\end{large}`. Observe el resultado. Lleve a cabo la misma tarea con las diferentes familias y perfiles.
- [5.3] Considere un documento que contenga un texto como el siguiente:
... texto ... \itshape queremos llamar la atención sobre este texto
principal \upshape lindo y sencillo
Compíelo para comprobar el resultado. Ponga una coma después de *principal* y compíelo de nuevo. Introduzca después de *principal* la corrección itálica `\textit{}` y compile de nuevo con coma y sin coma. ¿Observa las diferencias?
- [5.4] Escriba un párrafo utilizando al menos cinco colores de los reconocidos cuando se activa la opción `usenames`. Recuerde que si el documento de salida no es PostScript necesitará activar también la opción `dvipsnames`.
- [5.5] El documento `ejercicio5.5.ps` contiene texto en varios colores. Intente reproducirlo lo más aproximadamente posible; en el archivo `ejercicio5.5.tex` podrá encontrar la solución.

PARA SABER MÁS

- Las imprentas profesionales utilizan impresoras y fuentes PostScript, la mayoría diseñadas por empresas especializadas para su comercialización. Por otra parte, casi todas las impresoras PostScript tienen residentes 35 fuentes estándar, las conocidas como «fuentes PostScript de Adobe». L^AT_EX puede utilizar todas estas fuentes, siempre que estén instaladas en su computador (véase la sección II.1.4.4 para más detalles).
- Los colores se pueden identificar, además de por su nombre, por las combinaciones de otros colores básicos de los que están compuestos o por sus propiedades. Esto da lugar a los diferentes modelos de color, que se comentan en la sección II.3.1.
- Además de los 68 colores predefinidos, es posible definir y utilizar nuevos colores. Los comandos necesarios para ello se encuentran recogidos también en II.3.1.
- El fondo de las páginas es usualmente blanco, pero en ciertas ocasiones (por ejemplo, si estamos preparando una presentación electrónica) interesa que el color del fondo sea distinto. En el apartado II.3.1.3 se explica cómo conseguirlo.
- El uso de los colores en las tablas (que se explican en la lección 14) requiere un tratamiento especial, pues en este caso es interesante también poder colorear las diferentes casillas de la tabla. Para ello se necesita el paquete `colortbl`, que se explica en el apartado II.3.3.6.

Libros y artículos



OBJETIVOS

- Conocer las dos clases de documento fundamentales de L^AT_EX.
- Estructurar correctamente un texto.
- Incluir diversos elementos usuales: autor, título, apéndices, etc.
- Incluir el índice general.
- Conocer las principales opciones de las clases de documento y conocer sus efectos.

EN las lecciones anteriores hemos aprendido a escribir texto básico, con distintos tipos de letra y párrafos, así como algunos signos ortográficos importantes. Ya es hora de introducir una estructura en el documento. Todos los documentos están estructurados, es decir, poseen unos bloques bien diferenciados y caracterizados. Un libro, por ejemplo, suele iniciarse con un prólogo, le sigue el índice general (y, eventualmente, otros índices) y suele ser dividido en capítulos. A su vez, los capítulos están normalmente divididos en secciones, éstas en subsecciones, y así sucesivamente. Además, el libro puede poseer, usualmente en su parte final, apéndices, anexos, uno o varios índices (terminológico, onomástico, etc.) y la bibliografía.

En esta lección aprenderemos a utilizar este tipo de estructuras con L^AT_EX, excepto el tratamiento de la bibliografía y los índices terminológicos. El tema de la bibliografía será abordado en la lección 15, y posteriormente ampliado en el capítulo 2 de segunda parte; en ese capítulo será descrita también la forma de producir índices terminológicos con L^AT_EX.

Además de ello, o precisamente por eso mismo, vamos a introducir un cambio radical: abriremos la posibilidad de cambiar nuestro «preámbulo estandarizado», introducido en la lección 2, página 18. La razón es muy simple: las estructuras disponibles en un documento vienen determinadas por lo que se denomina, en terminología de L^AT_EX, las *clases de documento*, y la clase de documento que deseamos manejar se especifica en el preámbulo.

§6.1 Las clases de documento

EFFECTIVAMENTE, la clase de documento se determina en el preámbulo, más exactamente en la primera línea del código fuente. Para determinar dicha clase se recurre a un único comando, quizás el más importante de todo el documento; se trata, concretamente, del comando

```
\documentclass[Opciones]{NombreClase}
```

El argumento *NombreClase* especifica la clase de documento. Existen dos clases de documento fundamentales: *book* y *article*; la primera se concibe para la escritura de libros, mientras que el objetivo de la segunda es la escritura de documentos más breves que se asocian, en términos generales, a un artículo científico (y es la que figura en el preámbulo estándar para estas lecciones, hasta ahora manejado).

Otras dos clases de documento estándar son *report* (informe) y *proc* (proceedings, actas), derivadas, respectivamente, de *book* y *article*; son prácticamente iguales a éstas aunque con *Opciones* distintas. La distribución estándar de L^AT_EX incluye otras dos clases: *letter*, diseñada para la escritura de cartas, y *slides*, cuyo objetivo es la creación de transparencias y que, hoy en día, está cayendo en desuso en favor de las herramientas que permiten crear presentaciones electrónicas. Ninguna de estas dos últimas clases de documento será tratada en este libro (el lector interesado puede consultar [11]).

Existen muchas otras clases de documento que no forman parte de la distribución estándar de L^AT_EX: entre las más conocidas están las de la American Mathematical Society, pero, actualmente, muchas casas editoriales internacionales poseen sus propias clases de documento para uso de sus autores.

Todo ello muestra que es en estas clases de documento, cuyas acciones y comandos se encuentran definidos en los archivos de extensión *cls*, donde se configura el aspecto general y la estructura de un documento cualquiera.

El argumento *Opciones* permite seleccionar, para la clase elegida *NombreClase*, algunas de las opciones que dicha clase posee. Distintas clases pueden tener distintas opciones disponibles y distintas opciones elegidas internamente por defecto (es precisamente en estas opciones por defecto en lo que se distinguen *book* de *report*, por ejemplo). En el preámbulo estándar hemos manejado la opción *a4paper* que, claramente, indica el tamaño del papel, así como la opción *11pt*, que indica el tamaño de los tipos usuales, es decir, el tamaño especificado por la declaración *\normalsize* (véase la página 44). El apartado 6.6 contiene la descripción de algunas, las más utilizadas, de estas *Opciones*.

He aquí, finalmente, algunos ejemplos de declaración de la clase de documento con opciones diferentes:

```
\documentclass[a4paper,11pt]{book}
\documentclass[twocolumn]{article}
\documentclass[a4paper,12pt,draft]{article}
\documentclass{letter}
\documentclass[letterpaper,12pt]{letter}
\documentclass[a3paper,12pt]{book}
```

§6.2 Las unidades de estructura fundamentales

Las clases de documento básicas, `article` y `book` (y también `proc` y `report`), poseen una serie de comandos para dar estructura al documento: son los comandos que dividen el texto en unidades, como capítulos, secciones, etc. El cuadro 6.1 indica las unidades disponibles para cada una de las clases y la jerarquía natural que poseen: un capítulo contiene secciones, cada sección contiene subsecciones, etc. En este cuadro se indica el comando que introduce cada tipo de unidad y el nombre que le asignaremos a ésta (que proviene de la traducción, un tanto «literal», del nombre del comando).

Nombre	Clase <code>article</code>	Clase <code>book</code>
Parte	<code>\part</code> (optativa)	<code>\part</code> (optativa)
Capítulo		<code>\chapter</code>
Sección	<code>\section</code>	<code>\section</code>
Subsección	<code>\subsection</code>	<code>\subsection</code>
Subsubsección	<code>\subsubsection</code>	<code>\subsubsection</code>
Parágrafo	<code>\paragraph</code>	<code>\paragraph</code>
Subparágrafo	<code>\ subparagraph</code>	<code>\ subparagraph</code>

Cuadro 6.1: Jerarquía de las unidades de estructura según la clase de documento

Se observan en el cuadro citado dos características importantes. La primera es que las dos clases de documento comparten las mismas unidades de estructura con la excepción del capítulo, que sólo está disponible en `book`. La segunda es que la unidad más amplia (la parte) es optativa. A decir verdad, ninguna unidad es obligatoria, aunque si, por ejemplo, introducimos una subsección que no esté dentro de una sección se producirán algunos efectos indeseables (en particular, sobre la numeración de estas unidades), y esta práctica debe considerarse como una forma errónea de estructurar un documento. No caemos en una mala práctica si no incluimos la unidad parte (no todos los libros, por ejemplo, suelen estar divididos en estos grandes bloques, aunque sí lo está éste que usted se encuentra leyendo).

La sintaxis de `LATEX` para introducir estas unidades de estructura es la siguiente:

<code>\part [TextoToc] {Título}</code>	<code>\part*{Título}</code>
<code>\chapter [TextoToc] {Título}</code>	<code>\chapter*{Título}</code>
<code>\section [TextoToc] {Título}</code>	<code>\section*{Título}</code>
<code>\subsection [TextoToc] {Título}</code>	<code>\subsection*{Título}</code>
<code>\subsubsection [TextoToc] {Título}</code>	<code>\subsubsection*{Título}</code>
<code>\paragraph [TextoToc] {Título}</code>	<code>\paragraph*{Título}</code>
<code>\ subparagraph [TextoToc] {Título}</code>	<code>\ subparagraph*{Título}</code>

Como se observa, la sintaxis de todos estos comandos es idéntica en cada columna. A continuación se detallan los distintos elementos de esta sintaxis para los comandos sin asterisco.

- El argumento *Título* aparecerá como el título de la unidad especificada por el comando. El *Título* forma parte del encabezamiento (o título en un sentido más amplio) de la unidad. Este encabezamiento es compuesto por L^AT_EX de distintas formas, según se trate de una unidad u otra. Lo usual es, naturalmente, que a inferior nivel en la jerarquía, es decir, a menor amplitud de la unidad, el encabezamiento sea más discreto: tipos más pequeños, sin blancos de separación vertical, etc. Se pueden tener así desde grandes títulos, como en el caso de los \chapter, hasta títulos en línea o titulillos, como en los \paragraph.

Normalmente las unidades de estructura son numeradas automáticamente y el *Título* estará precedido por el número de orden de la unidad, si bien este comportamiento depende de varios factores que se indicarán en su momento. Este número (de capítulo, sección, etc.) puede ser «representado» (impreso) de múltiples formas (como cualquier otro contador de L^AT_EX, véase §17.1) y sirve como elemento identificador de la unidad, para que pueda ser referenciada en otros lugares del texto (como aprenderemos en la lección 8).

En el caso de \part y \chapter el encabezamiento incluye el número precedido por los antetítulos ‘Parte’ y ‘Capítulo’, respectivamente (con la opción spanish de babel).

- El argumento optativo *TextoToc* es el texto que deseamos que aparezca en el índice general para hacer referencia a la unidad en cuestión; si este argumento no aparece L^AT_EX escribirá el *Título* en dicho índice (véase §6.5). Además, algunos de estos comandos escriben *TextoToc*, o, en su defecto, *Título*, como parte de la cabecera de las páginas (en la parte conocida técnicamente como *folio*, en este caso un folio descriptivo). Para conocer exactamente qué comandos realizan esta tarea y en qué condiciones, debemos esperar hasta la segunda parte de este libro.

Como se aprecia, el argumento *TextoToc* puede ser útil como alternativa a *Título*, cuando este último es excesivamente largo. Con ser ventajosa esta forma de abreviar el texto que aparece en la cabecera, para algunos usuarios, sin embargo, puede resultar rechazable que también se refleje en el índice general, es decir, que la entrada en dicho índice no coincida exactamente con el *Título* de la unidad. En este caso existe una vía distinta para alterar el texto de la cabecera, sin modificar la entrada en el índice general. Para ello iniciaremos una unidad sin el argumento optativo *TextoToc* y recurriremos, en la forma explicada a continuación, al comando adecuado entre los comandos siguientes:

```
\chaptermark{TextoCab}  
\sectionmark{TextoCab}  
\subsectionmark{TextoCab}
```

Si la cabecera que deseamos alterar es iniciada por un \chapter necesitaremos el comando \chaptermark y de forma análoga para las otras dos unidades. El argumento *TextoCab* es el texto que queremos que aparezca en la cabecera de las páginas. Estos comandos deben ser situados inmediatamente a continuación del comando que inicia la unidad correspondiente. Existen sin embargo situaciones en las que el comportamiento resultante no es el esperado; más concretamente, si, por ejemplo, al introducir el comando \sectionmark inmediatamente después de \section, la cabecera no cambia, debemos introducir \sectionmark

antes y después de \section. Esta situación sólo puede darse en la misma página en la que se inicia la unidad y en las siguientes condiciones:

- con \section y \sectionmark en la clase book (sin opción oneside, véase §6.6);
- con \subsection y \subsectionmark en la clase article (con opción twoside y estilo de página headings, véase el apartado 7.1).

Los comandos de estructura con asterisco difieren de sus contrapartidas sin asterisco en dos aspectos: nunca numeran la unidad y no introducen nada en el índice general ni en la cabecera de las páginas (por esa razón no poseen el argumento optativo *TextoToc*). El ejemplo 6.1 ilustra el uso de los comandos con y sin asterisco: observe que las unidades iniciadas con asterisco no sólo no escriben el número de la unidad, sino que no alteran la forma de contar las unidades.

EJEMPLO 6.1

```
\documentclass{book}
\usepackage[english]{babel}
\begin{document}

\chapter*{Preface}
Gentle Reader: This is a handbook about TeX...
.....(nueva página)....
```

```
\chapter{The Name of the Game}
English words like \textquotel left technology' stem for a Greek root...
```

```
\chapter{Book Printing versus Ordinary Typing}
When you first started using a computer terminal\...
```

(Tomado de ‘‘The \TeX{} book’’ por Donald E. Knuth)

Preface

Gentle Reader: This is a handbook about TeX...

Chapter 1

The Name of the Game

English words like ‘technology’ stem for a Greek root...

Chapter 2

Book Printing versus Ordinary Typing

When you first started using a computer terminal,...

Unidades de estructura con y sin asterisco (las cajas simulan páginas). Observe que el nombre ‘Capítulo’ ha cambiado a ‘Chapter’, porque se usa la opción english de babel

Hay algunos comportamientos de estos comandos de estructura que dependen de las opciones elegidas para la clase de documento. Puesto que cada clase posee unas opciones por defecto, podríamos describir aquí las acciones adicionales que se toman de esta manera, por defecto, pero esta descripción se pospone al apartado 6.6. Una pequeña experiencia mostrará, por ejemplo, que un capítulo (\chapter, por tanto, necesariamente en la clase book) se inicia en una nueva página, que siempre es una página impar (mientras que una opción de \documentclass no cambie este comportamiento), lo que significa que puede efectuar uno o dos saltos de página automáticamente.

§6.3 Portada, prólogos y apéndices

Un libro posee muchas partes más o menos usuales: antepartida, portada, página de derechos, dedicatoria, prólogo (entre las iniciales) y apéndices, anexos e índices alfabéticos (entre las finales). Un artículo posee muchas menos pero, desde luego, siempre incluye un autor y un título,

y, muy frecuentemente, un resumen (elemento que no aparece en los libros). Es usual que estas unidades de estructura sean tratadas de forma diferente a las que constituyen el cuerpo central del documento; L^AT_EX no olvida esta buena costumbre y dispone de varias facilidades para llevar a cabo esta tarea.

Título y autor

L^AT_EX trata el título del documento junto al autor y cualquier información complementaria sobre éste como un todo; el conjunto constituye la «página del título» (que no tiene por qué ser una auténtica página completa).

Esta página del título se construye a partir de los distintos elementos que la integran y que son declarados con los comandos descritos a continuación. Estos comandos no imprimen ningún texto por sí mismos; su posición más adecuada en el documento fuente es el preámbulo.

```
\title{Título}
```

Este comando declara al argumento *Título* como título del documento. En *Título* es posible usar saltos de línea \\, si es conveniente (lo que puede ocurrir si el título es largo; véase el término ‘Título’ en [60] para algunas indicaciones sobre el lugar adecuado para separar palabras en un título); no es posible, sin embargo, incluir una línea en blanco o un comando \par (véase el ejemplo 6.2).

```
\author{Autor1 \and Autor2 \and ...}
```

Así se declaran los autores del documento. El argumento *Autor1* es un autor; si hay más de un autor, cada uno de ellos debe ser separado por el comando \and. En cada uno de los textos *Autor1*, *Autor2...* podemos utilizar el salto de línea \\.

```
\date{FechaTexto}
```

Este comando nos sirve para incluir cualquier comentario adicional en la «página del título». Su función original es la de incluir la fecha de redacción del documento, pero el argumento *FechaTexto* puede incluir cualquier texto y el salto \\. Si este comando se omite se imprimirá como fecha la que posea el computador en el momento de la compilación. Si lo que deseamos es que no aparezca ningún comentario ni fecha, entonces debemos incluir este comando con argumento vacío, es decir, \date{}. El comando \today puede sernos útil a la hora de generar el argumento *FechaTexto* (v. pág. 15).

```
\thanks{Texto}
```

En cualquiera de los argumentos de los tres comandos anteriores (*Título*, *Autor1*, *Autor2...*, *FechaTexto*) se puede incluir el comando \thanks, lo que produce una nota al pie de página. Esto suele utilizarse, como el nombre del comando indica, para los agradecimientos. Si este comando no está al final de una línea o no está seguido de otro comando, conviene escribir el comando _ a continuación de él (v. pág. 24).

Finalmente

```
\maketitle
```

imprimirá la «página del título» teniendo en cuenta para ello todo lo especificado por los comandos anteriores. Este comando debe aparecer una única vez (las repeticiones son ignoradas), con el preámbulo finalizado, es decir, después de `\begin{document}` (por lo común, inmediatamente a continuación) y también después de los comandos antes descritos.

La «página del título» no es necesariamente toda una página; esto depende de cuál de las opciones `titlepage|notitlepage` haya sido seleccionada (a este respecto véase el apartado 6.6).

Existe una segunda forma de generar una «página de título»: con el uso del entorno

```
\begin{titlepage}
TextoConFormato
\end{titlepage}
```

Optar por el uso de este entorno tiene una gran desventaja frente a `\maketitle`: el que escribe es completamente responsable del diseño de toda la página (en este caso se trata de una verdadera página, independientemente de la opción `titlepage|notitlepage` elegida). La ventaja de este entorno es que puede aparecer tantas veces como sea necesario, y, por tanto, podemos generar con él tantas páginas especiales como queramos; puede servirnos, pues, para crear una anteportada, una página de derechos, etc. En el entorno `titlepage`, *TextoConFormato* debe contener el texto que deseemos incluir en la página y los comandos necesarios para componerla completamente. Al igual que `\maketitle` este entorno realizará algunas tareas adicionales, que pueden consistir en uno o más saltos de página y la eliminación de cabeceras y pies de página y que, en todo caso, dependen de la clase de documento utilizada y de sus opciones.

Prólogo y resumen

En general, entenderemos aquí por prólogos cualquier unidad que aparezca en la parte inicial de un documento y que contenga suficiente texto; podríamos pensar en un prólogo, propiamente dicho, o en una introducción. Distinguimos este tipo de unidades de las páginas de derechos, dedicatorias o similares, que se adaptan mejor a ser introducidas en un documento L^AT_EX mediante el entorno `titlepage`. Lo característico de estas unidades de tipo prólogo es que pueden ser tipografiadas como si se tratase de un capítulo (o sección, en caso de que sea un documento de la clase `article`), salvo que no son numeradas. Por ello deben ser introducidas mediante un comando de estructura con asterisco (como el «Preface» del ejemplo 6.1; véase, sin embargo, a este respecto la explicación sobre la unidad declarada mediante `\frontmatter` en el apartado 6.4).

Los artículos suelen contener un ‘Resumen’ (en inglés ‘Abstract’ o ‘Summary’), una breve unidad que contiene, como su nombre indica, las ideas fundamentales que se desarrollan en el trabajo. Esta unidad se introduce en L^AT_EX mediante el entorno

```
\begin{abstract}
Texto
\end{abstract}
```

Esta unidad es más propia de artículos e informes, por lo que no está disponible en la clase `book` (aunque sí en la clase `report`). Naturalmente, la inclusión de este entorno es totalmente volun-

EJEMPLO 6.2

```
\documentclass[a4paper,11pt]{book}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[spanish]{babel}
\renewcommand{\shorthandsspanish}{}
\title{%
    Analyse des infinitiment petits\\
    pour l'intelligence des lignes courbes%\footnote{Considerado como el primer libro de texto de Cálculo Infinitesimal, contiene la primera aparición de la famosa regla Bernoulli.}\\
    \thanks{Considerado como el primer libro de Cálculo Infinitesimal, contiene la primera aparición de la famosa regla Bernoulli.}\\
    \thanks{Publicado por vez primera en el año 1696}
}
\author{%
    Guillaume François Antoine Marquis de l'Hospital\footnote{Guillaume François Antoine Marquis de l'Hospital, Jean Bernoulli\footnote{Jean Bernoulli}}\\
    and\\
    Jean Bernoulli\footnote{1667-1748. Actualmente se le considera el verdadero autor de esta obra. Como resultado de un extraño acuerdo comercial, Bernoulli enviaba sus resultados como lecciones para l'Hospital, lecciones que éste parecía haber recogido para publicar esta obra.}%
}
\date{Publicado por vez primera en el año 1696}
\begin{document}
\maketitle
...

```

¹Considerado como el primer libro de texto de Cálculo Infinitesimal, contiene la primera aparición de la famosa regla Bernoulli.

²1661-1701.
³1667-1748. Actualmente se le considera el verdadero autor de esta obra. Como resultado de un extraño acuerdo comercial, Bernoulli enviaba sus resultados como lecciones para l'Hospital, lecciones que éste parece haber recogido para publicar esta obra.

taria. La unidad se inicia con el antetítulo ‘Resumen’ (con `babel`, opción `spanish`) centrado. La forma de componer el contenido del entorno `abstract` depende también de las opciones `titlepage|notitlepage`.

Apéndices

Los apéndices aparecen en la parte final del documento y, a diferencia del prólogo y similares, suelen ser numerados, aunque, habitualmente, la numeración es distinta a la de capítulos y secciones. No incluimos bajo el concepto de apéndice unidades tan representativas y esenciales como la bibliografía y los índices alfabéticos, que poseen en L^AT_EX herramientas específicas para su construcción (véanse la lección 15 y el apartado PARA SABER MÁS). Así mismo, se pueden distinguir, en la técnica tipográfica, no en L^AT_EX, los conceptos de apéndice y anexo; este último es una unidad, normalmente de poca extensión, de carácter técnico, cuyo autor suele ser distinto al autor del documento en que se incluyen [60]. La declaración

```
\appendix
```

indicará que, a partir de ella, deseamos escribir los apéndices.

Cada uno de esos apéndices reales que queramos escribir, se introducirá como una unidad de estructura usual, de alto nivel, es decir: con `\chapter` en la clase `book`, y con `\section` en la clase `article`. La declaración `\appendix` cambia entonces la forma de tratar esas unidades:

- En la clase `book` cada capítulo llevará el antetítulo ‘Apéndice’ (en lugar de ‘Capítulo’, desde luego con la opción `spanish` de `babel`) seguido del número del apéndice, que ahora será impreso como una letra mayúscula: el primer `\chapter` numerado, después de `\appendix`, llevará por número el valor ‘A’, el siguiente ‘B’, etc.
- En la clase `article`, donde no es posible utilizar `\chapter`, cada sección tendrá una numeración en letras mayúsculas, pero no se imprimirá ningún antetítulo como ‘Apéndice’.

§6.4 Grandes unidades de estructura en la clase book

La clase `book` prevé tres grandes bloques en un libro: la unidad «frontal», la «principal» y la «posterior». La primera de ellas contendría todas las posibles «páginas de título» (portadilla, portada, página de derechos, dedicatoria, lema, etc.), los prólogos, el índice general, los índices de cuadros y figuras... un sinfín de posibilidades. La parte principal contendría el contenido del libro propiamente dicho y los apéndices. Finalmente la unidad posterior estaría ocupada por los anexos, bibliografía e índices alfabéticos.

Las siguientes declaraciones inician cada una de estas tres grandes unidades:

```
\frontmatter
```

```
\mainmatter
```

```
\backmatter
```

No es obligatorio utilizar estas unidades pero, como se observará por sus acciones, es conveniente hacerlo, ya que ayudan a la configuración de un libro siguiendo las tradiciones tipográficas y de composición más extendidas y adecuadas.

En la unidad iniciada por `\frontmatter` las páginas son numeradas en números romanos (en versalita, con la opción `spanish` de `babel`, y en minúscula sin ella). Además el comando `\chapter`, dentro de esta primera gran unidad, no imprime el número del capítulo ni el antetítulo ‘Capítulo’ (aunque sí incluye una línea en el índice general, véase §6.5). El resto de los comandos que inician las unidades de estructura (`\section`, etc.) no se comportan del mismo modo; así pues, estos últimos debemos usarlos con asterisco si no queremos que se imprima el número de la unidad (que además podría incluir el número de capítulo, lo que daría el número cero, algo realmente inaceptable desde un punto de vista estético).

Al iniciarse la unidad principal mediante la declaración `\mainmatter` la numeración de las páginas vuelve a comenzar. No ocurre lo mismo, sin embargo, con la declaración `\backmatter` que no altera dicha numeración. En ambas unidades los números de página se imprimen, por defecto, en su versión arábiga.

En la unidad que inicia `\backmatter` los comandos `\chapter`, `\section`, etc., se comportan exactamente de la misma forma que en la unidad frontal, descrita anteriormente.

§6.5 Crear el índice general

L índice general de un documento se compone a partir de las unidades de estructura en las que se ha dividido dicho documento. L^AT_EX proporciona, de forma muy sencilla, la construcción automática del índice general. Basta introducir, en el lugar donde queramos que se incluya este índice, el comando

`\tableofcontents`

El índice general se inicia con un título: ‘Índice general’, en la clase `book`, e ‘Índice’, en la clase `article` (siempre con opción `spanish` de `babel`).

Es fácil comprobar que se necesitan dos compilaciones para que efectivamente se escriba el índice general (o para que se actualice después de algún cambio en las unidades de estructura). El porqué de este extraño comportamiento, bastante usual en L^AT_EX, es que en la primera compilación el compilador escribe en un fichero auxiliar la información que debe aparecer en el índice, mientras que en la segunda utiliza dicho fichero auxiliar para construir de hecho dicho índice, con el formato adecuado. El fichero auxiliar al que nos hemos referido recibe el mismo nombre que el fichero que se está compilando y extensión `toc` (de «table of contents»).

Cada entrada en el índice general es generada por uno de los comandos de estructura sin asterisco; los comandos con asterisco no introducen absolutamente nada en el índice. El texto de cada entrada contiene el valor del argumento *Título* o el de *TextoToc*, si éste ha sido utilizado (véase §6.2); además, la entrada incluye, naturalmente, la numeración de la unidad, los típicos puntos conductores (aunque éstos no siempre aparecen) y el número de la página en la que se inicia la unidad a la que se refiere la entrada.

Sin embargo no todos los comandos de estructura (sin estrella) introducen una entrada en el índice general: en la clase `book` lo hacen `\part`, `\chapter`, `\section` y `\subsection`, mientras que en `article` lo hacen `\part`, `\section`, `\subsection` y `\subsubsection` (es decir, en am-

bos casos, las cuatro unidades más altas en la jerarquía, véase el cuadro 6.1). Naturalmente este comportamiento puede ser alterado por el usuario.

§6.6 Opciones básicas de la clase de documento

LAS opciones de la clase de documento determinan el comportamiento de muchos de los mandos estudiados en esta lección, así como de otras estructuras de L^AT_EX. Como ya hemos comentado, en este apartado nos restringiremos al estudio de las opciones más importantes; las restantes, así como más detalles sobre algunas de las aquí tratadas, se abordarán en la segunda parte.

Ya sabemos que las opciones se indican en el argumento optativo de \documentclass (el argumento denominado *Opciones*, véase la página 49). La sintaxis de dicho argumento consiste en una lista de opciones separadas por comas.

Es importante tener en mente la siguiente observación: una opción de la clase de documento *afecta a todos los paquetes cargados* que admitan dicha opción. Esto significa que aquellos paquetes, cargados mediante \usepackage, que reconozcan como suya una opción incluida en el comando \documentclass la tomarán como activada para ellos mismos. Por ejemplo, los paquetes hyperref y graphicx (véanse las lecciones 8 y 9) poseen la opción draft, por lo que el código

```
\documentclass[draft]{article}
\usepackage{graphicx}
\usepackage{hyperref}
```

es equivalente a:

```
\documentclass[draft]{article}
\usepackage[draft]{graphicx}
\usepackage[draft]{hyperref}
```

Observe que esto es una ventaja, pero también puede suponer un inconveniente cuando se desea incluir una opción para la clase de documento y no para uno o más paquetes. Para algunos paquetes cada opción posee una opción opuesta, lo que, en tales casos, resuelve el problema.

Algunas clases de documento y algunos paquetes no reconocen ciertas opciones, simplemente no las tienen declaradas como tales, no están diseñados para ello. El compilador L^AT_EX emitirá un mensaje de precaución cuando ni la clase de documento ni ninguno de los paquetes cargados reconozcan una opción.

Una clase de documento posee siempre opciones por defecto; no es preciso especificar éas en el comando \documentclass. Así, por ejemplo, el código

```
\documentclass[10pt]{article}
```

es equivalente a:

```
\documentclass{article}
```

ya que 10pt es la opción por defecto (en cuanto al tamaño de los tipos) de la clase article.

En el cuadro 6.2 se especifican las opciones básicas disponibles en las clases de documento usuales, así como las que, entre aquéllas, toman por defecto. Lo importante para el buen manejo

	article	proc	book	report
10pt	D	D	D	D
11pt	×	×	×	×
12pt	×	×	×	×
letterpaper	D	D	D	D
legalpaper	×	×	×	×
executivepaper	×	×	×	×
a4paper	×	×	×	×
a5paper	×		×	×
b5paper	×		×	×
final	D	D	D	D
draft	×	×	×	×
oneside	D	D	×	D
twoside	×	×	D	×
openright			D	×
openany			×	D
onecolumn	D		D	D
twocolumn	×	D	×	×
notitlepage	D	D	×	×
titlepage	×		D	D

Cuadro 6.2: Opciones básicas de las clases de documento. «D» significa opción por defecto; «×» significa que la opción está disponible; un espacio en blanco significa no disponibilidad (si se usa la opción en tales casos puede producirse un error o un simple mensaje de precaución). La definición de las acciones realizadas por las opciones suele estar contenida en los archivos de extensión `.clo`

de las opciones es, en primer lugar, comprender bien en qué consiste cada una de ellas, para después, en segundo lugar, consultando el cuadro 6.2, incluir la opción adecuada que, cambiando el comportamiento por defecto de la clase, proporcione el resultado que deseamos.

A continuación se detallan las distintas opciones y su significado. Las opciones van agrupadas según el comportamiento al que afectan y separadas, en cada grupo, por una barra vertical «|». Las opciones que están agrupadas son mutuamente excluyentes, es decir, que no debemos usar dos del mismo grupo simultáneamente.

10pt|11pt|12pt

Especifican el tamaño del texto normal (el determinado por el comando `\normalsize`) dado en «puntos» (véase la página 143). En el cuadro 6.3 se puede observar la comparación de los distintos tamaños de los tipos (expuestos en el apartado 5.4) para las tres opciones anteriores.

10pt	\tiny		\scriptsize	\footnotesize	\small	\normalsize
11pt		\tiny		\scriptsize	\fotnotesize	\small
12pt		\tiny		\scriptsize		\fotnotesize
1em=	6.8039pt	7.33154pt	7.97028pt	8.49792pt	9.24774pt	9.99756pt
10pt		\large	\Large	\LARGE	\huge	\Huge
11pt	\normalsize	\large	\Large	\LARGE	\huge	\Huge
12pt	\small	\normalsize	\large	\Large	\LARGE	\huge \Huge
1em=	10.88788pt	11.74713pt	13.70703pt	16.23491pt	19.18753pt	22.72049pt

Cuadro 6.3: Tamaños relativos de los tipos según la opción de la clase de documento. Los tamaños que se sitúan en la misma columna son iguales. La longitud de la última fila es la medida de la unidad em (véase la página 143) seleccionada la opción T1 del paquete fontenc

letterpaper|legalpaper|executivepaper|a4paper|a5paper|b5paper

Mediante una opción de este grupo indicamos el tamaño de papel que utilizaremos (observe que el estándar en España es el tamaño A4, que no corresponde a la opción por defecto). L^AT_EX se encarga internamente de asignar valores a un buen número de parámetros, algunos de los cuales se exponen en el apartado 7.2 a fin de adaptar el texto impreso al tamaño de papel seleccionado por la opción. Las dimensiones de los tamaños de papel asociados a las distintas opciones se indican en el cuadro 6.4.

letterpaper	8,5 in × 11 in (215,9 mm × 279,4 mm)
legalpaper	8,5 in × 14 in (215,9 mm × 355,6 mm)
executivepaper	7,25 in × 10,5 in (215,9 mm × 266,7 mm)
a4paper	210 mm × 297 mm
a5paper	148 mm × 210 mm
b5paper	176 mm × 250 mm

Cuadro 6.4: Tamaños estándar de papel

final|draft

Cuando, al final de línea, o, más generalmente, dentro de una caja horizontal (véase la lección 18), T_EX no sabe dividir siládicamente la última palabra, se produce un mensaje del tipo «Overfull \hbox...»: la palabra no será dividida y, por tanto, la línea de texto será más ancha de lo normal. Si se ha seleccionado la opción draft dichas líneas serán marcadas con un rectángulo negro en el margen derecho, con el fin de hacerlas visualmente detectables con facilidad. Más generalmente, el término «draft» debe ser interpretado como «borrador», lo que sugiere, y así es en efecto, en muchos casos, que esta opción (referida a un ámbito más amplio que el de las clases de documento)

aporta facilidades de manipulación de versiones no definitivas de nuestros documentos. La opción final, opuesta a `draft`, no marca las líneas en las que existe un `Overfull`.

oneside|twoside

Estas opciones deciden si el documento se preparará para ser impreso a una o dos caras, respectivamente. Están en relación con los llamados *estilos de página* y afectan a los márgenes del texto y a la cabecera de las páginas (véanse los apartados 7.1 y 7.2). Es muy importante introducir en este momento un poco de nomenclatura: en L^AT_EX se habla de páginas «a derecha» y «a izquierda», no tanto de páginas impares y pares. Cuando la opción elegida es `twoside` ambos conceptos coinciden: así las páginas situadas a la derecha son siempre las impares. Cuando la opción elegida es la de impresión a una sola cara (`oneside`), entonces todas las páginas son consideradas a derecha; en tal caso, cualquier indicación sobre páginas a izquierda es irrelevante.

openright|openany

La opción `openright` especifica que todos los capítulos comenzarán en una página «a derecha» (en el sentido indicado en el párrafo anterior); esto significa que, en algunos casos, al iniciarse un capítulo (con `\chapter`) se puede producir un doble salto de página. Con la opción opuesta, `openany`, un capítulo se iniciará siempre en una nueva página que podrá ser a derecha o izquierda, indistintamente. Observe que estas opciones se refieren sólo al comportamiento del comando `\chapter` (de ahí que no estén disponibles para las clases `article` y `proc`, véase el cuadro 6.2).

onecolumn|twocolumn

Determinan si el texto será compuesto a una o dos columnas, respectivamente.

notitlepage|titlepage

Con la opción `titlepage` activada, el comando `\maketitle` ubicará la «página del título» en una sola página real e independiente (véase el comentario sobre esta «página del título» en la página 54) y el entorno `abstract` se comportará del mismo modo respecto a su contenido. Con la opción `notitlepage` el texto usual del documento seguirá al título y al `abstract` en la misma página que éstos.

Ejercicios

- 6.1** Escoja un libro de su biblioteca personal: si es posible uno científico, un ensayo o similar, que posea una estructura suficientemente complicada.

Intente reproducirlo: comenzando con los capítulos, secciones, etc. (naturalmente basta que escriba unas líneas del comienzo de cada una de estas unidades). Continúe incluyendo título, autor, fecha, etc. (con `\maketitle` y sin intentar reproducir el formato de la portada).

- 6.2** Intente reproducir la página de derechos del mismo libro (copyright, editor, fecha, impresor, lugar de edición, etc.) incluyéndola en un entorno `titlepage`. Haga algo similar con una página de dedicatoria (si no la tiene, invéntela).

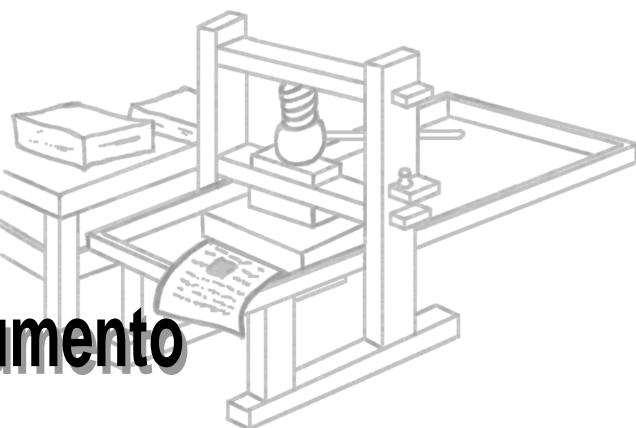
- 6.3** Siguiendo con el mismo modelo, introduzca los tres grandes bloques de la clase `book`. Observe los cambios.

- 6.4 Cree el índice general. En algunas unidades de estructura introduzca el argumento optativo de los comandos que las inicián (el argumento que hemos llamado *TextoToc*). Observe los cambios que se producen en la cabecera de las páginas y en el índice general.
- 6.5 Sobre el documento realizado a partir del ejercicio 6.1 cambie la clase book por report. Observe los cambios. Introduzca las opciones twoside y openright: ¿recupera exactamente la misma salida que en el caso de la clase book?
- 6.6 Simule un artículo (puede utilizar el mismo código generado en los ejercicios anteriores, cambiando la clase de documento y eliminando los \chapter). Introduzca todos los niveles de estructura (hasta \subparagraph). Observe cómo L^AT_EX compone los titulillos de las unidades inferiores. ¿Qué unidades se han numerado? Incluya el índice general: ¿qué unidades escribieron una entrada en este índice?

PARA SABER MÁS

- ▶ Los antetítulos ‘Parte’ para \part, ‘Capítulo’ para \chapter, etc. pueden modificarse; véanse las secciones II.1.3 y II.2.1.
- ▶ La numeración de las unidades de estructura es importante: ¿para qué unidades de estructura se imprime el número de orden?, ¿cómo se imprime?; véase II.2.1.
- ▶ Sobre cómo se gestionan los textos que van a la cabecera de la página y, en particular, el porqué de los extraños comportamientos de \sectionmark, en II.2.4.1.
- ▶ Cómo decidir qué unidades de estructura incluirán automáticamente una entrada en el índice general y cuáles no, también en II.2.1.
- ▶ Cómo incluir manualmente entradas para las unidades iniciadas con un comando con asterisco y, más generalmente, poder incluir texto arbitrario directamente en dicho índice general, véase II.2.2.1.
- ▶ Sobre otras *Opciones* en la declaración de la clase de documento, véase II.2.6.
- ▶ Una de las tareas más arduas en la escritura de textos largos, como un libro, es la creación de un índice terminológico, onomástico o topográfico. L^AT_EX y el programa MAKEINDEX constituyen también una eficacísima herramienta para ello. En II.2.7 encontrará mucha de la información necesaria.
- ▶ Son muchos los parámetros que L^AT_EX maneja para determinar el diseño de un documento y de sus páginas, en la sección II.2.5 encontrará una descripción detallada de ellos.
- ▶ Los libros suelen ser extensos. Si cada vez que queremos compilar para verificar lo último escrito hemos de soportar la compilación, digamos, del centenar de páginas anteriores, ya correctas, el trabajo se hace tedioso y no exento de problemas a la hora de la manipulación y control del código fuente. La solución es separar un documento largo en trozos, por ejemplo un libro en capítulos, que se compilarán por separado. Para conocer las posibilidades de esta técnica, véase II.2.3.

Paginar un documento



OBJETIVOS

- Familiarizarse a nivel básico con los estilos de página.
- Modificar los principales parámetros que determinan el aspecto visual de una página.
- Modificar la paginación final: insertar espacios verticales y horizontales, forzar un salto de página, etc.

LA clase de documento seleccionada (véase la lección 6) instruye a L^AT_EX para que éste determine el aspecto y la estructura globales del documento. Una de las cuestiones de las que L^AT_EX se ocupa es del diseño de las páginas. Algunos elementos de este diseño cambiarán con la clase de documento y sus opciones. Una página está compuesta, en una primera aproximación, por los siguientes elementos:

- El encabezamiento (también llamado cabecera).
- El cuerpo central (el texto propiamente dicho).
- El pie, que no debe confundirse con la zona reservada a las notas a pie de página, que se consideran parte del cuerpo y se ubican en la parte inferior del mismo.

En la mayoría de las páginas de este libro la cabecera contiene el número de página y el título de la sección, lección o capítulo, todo enmarcado en un rectángulo sombreado; sin embargo, en esta página en concreto el encabezamiento es vacío. El cuerpo central de una página consiste en todo lo que hay entre la cabecera y el pie: el texto principal, notas a pie de página, figuras y tablas. El pie puede ser utilizado, por ejemplo, para la numeración de las páginas, si preferimos que ésta no se lleve en la cabecera. Los gráficos incluidos en la figura 7.1 explican los elementos de una página y los parámetros que controlan su diseño. En esta lección aprenderemos a modificar algunos de estos parámetros para que la paginación final del documento que estamos componiendo quede a nuestro gusto. Hemos de advertir que los problemas que pueden aparecer a la hora de dar el último «toque» al aspecto de un documento pueden ser muy complicados y que muchas veces es aconsejable no modificar los parámetros que L^AT_EX tiene por defecto para el diseño de las páginas.

§7.1 Estilos de página

El contenido del encabezamiento y del pie de una página viene determinado por los llamados estilos de página. En L^AT_EX existen varios estilos de página predeterminados que responden a distintos gustos o necesidades. Los más importantes de estos estilos son:

- `plain`) la cabecera está vacía y el pie contiene únicamente el número de página centrado;
- `empty`) la cabecera y el pie están vacíos;
- `headings`) la cabecera contiene el número de página y un texto determinado por la clase de documento (normalmente por los comandos que inician las unidades de estructura, véase la lección 6); el pie de página está vacío.

Las siguientes declaraciones permiten elegir un estilo:

<code>\pagestyle{Estilo}</code>	<code>\thispagestyle{Estilo}</code>
---------------------------------	-------------------------------------

Mientras que `\pagestyle{Estilo}` determina el *Estilo* hasta que aparezca de nuevo otro comando `\pagestyle`, el estilo elegido con `\thispagestyle` sólo se aplicará a la página que se esté «componiendo» en el momento de ejecutar dicho comando.

El estilo que, por defecto, selecciona la clase `book` es `headings`, mientras que la clase `article` selecciona `plain`. Cualquiera de estas determinaciones del estilo de página puede ser reemplazada mediante el uso de las declaraciones anteriores `\pagestyle` y `\thispagestyle`. Por ejemplo, si estamos usando la clase de documento `article` y queremos que nuestro documento lleve las páginas sin numerar, incluiremos el comando `\pagestyle{empty}` en el preámbulo. Pudiera ocurrir que, a pesar de haber incluido la instrucción anterior, L^AT_EX todavía numere la primera página. Para hacer que la primera página no se numere debemos incluir `\thispagestyle{empty}` en la primera página, con la precaución de incluir esta instrucción inmediatamente detrás del comando `\maketitle`, si es que éste también está en la primera página (véase la página 53, donde se explica el uso y función de `\maketitle`).

§7.2 Parámetros de una página

Una vez elegida la clase de documento que vamos a utilizar para componer nuestro texto e iniciada la escritura del mismo, puede que queramos modificar la *mancha* final del documento, que por defecto es pequeña, para aprovechar más el papel, o puede que queramos trasladar los márgenes de salida de la misma. El tamaño (anchura y altura) y posición (márgenes horizontal y vertical de salida) de la mancha del texto son longitudes que están fijadas por defecto por la clase de documento, y por sus opciones, que estamos utilizando. Anchura y altura del texto están fijadas, respectivamente, mediante las longitudes

<code>\textwidth</code>	<code>\textheight</code>
-------------------------	--------------------------

y los márgenes de salida horizontal y vertical están fijados mediante las longitudes

<code>\hoffset</code>	<code>\voffset</code>
-----------------------	-----------------------

Estas longitudes están ilustradas en la figura 7.1 y explicadas en la lista que cierra este apartado. Utilizando los comandos explicados en el apartado 17.2 podemos modificar los valores que la clase de documento fija por defecto para estas y otras longitudes que controlan el diseño de las páginas. Así, por ejemplo, mediante las líneas de código

```
\textwidth 15cm  
\hoffset -1cm
```

incluidas en el preámbulo de un documento, se fija la anchura del texto en 15 cm (en términos absolutos) y se corre hacia la izquierda 1 cm el margen horizontal de salida. Si, por ejemplo, estamos utilizando la clase de documento `article` con la opción de papel `a4paper`, el valor por defecto de `\textwidth` es de 12 cm, y todas las demás longitudes fijadas por defecto para el diseño de la página están calculadas para que la mancha obtenida esté centrada con respecto al papel. En una situación práctica, si queremos producir un documento con una anchura de texto de 16 cm las líneas de código

```
\addtolength{\textwidth}{4cm}  
\addtolength{\hoffset}{-2cm}
```

ofrecen una solución satisfactoria para cambiar las dimensiones del texto, ya que con estas instrucciones se preserva que el texto todavía esté centrado con relación al papel. El lector ya habrá entendido lo que está pasando en esta segunda opción que proponemos, y que recomendamos encarecidamente, para modificar las longitudes de una página. Nuestros comentarios para `\textwidth` y `\hoffset` se aplican de forma similar para `\textheight` y `\voffset`.

La lista que sigue contiene explicaciones más detalladas sobre las longitudes comentadas anteriormente.

`\evensidemargin` Cuando está en uso la opción `twoside` (véase la página 61), esta longitud se refiere a las páginas a izquierda (pares) e indica la distancia que separa el «margen horizontal de salida» (definido más adelante, en el apartado sobre `\hoffset`) del texto (véase también, a este respecto, `\oddsidemargin`). Obsérvese que, una vez encuadrado el documento, esta longitud se refiere al margen exterior (de las páginas a izquierda). Si no se utiliza la opción `twoside` esta longitud es irrelevante.

`\hoffset` Esta longitud define el margen izquierdo de salida horizontal o «borde izquierdo de impresión». Los «visores» o programas de impresión de ficheros DVI definen el ángulo superior izquierdo del espacio reservado para la impresión en un punto que no suele coincidir con el ángulo superior izquierdo del papel. Lo más habitual es que este punto de referencia se sitúe una pulgada a derecha del borde izquierdo y a la misma distancia del borde superior del papel. El valor de `\hoffset` se añade a esta distancia horizontal, determinando, así, la coordenada horizontal del punto de referencia, lo que hemos llamado «borde izquierdo de impresión». El valor de `\hoffset` puede ser positivo o negativo, lo que supone, respectivamente, un incremento o una disminución de la distancia horizontal al borde izquierdo del papel que, por defecto, define el «visor».

La asignación del valor de `\hoffset` debe ser colocada en el preámbulo del documento, y puede, incluso, preceder al comando `\documentclass`.

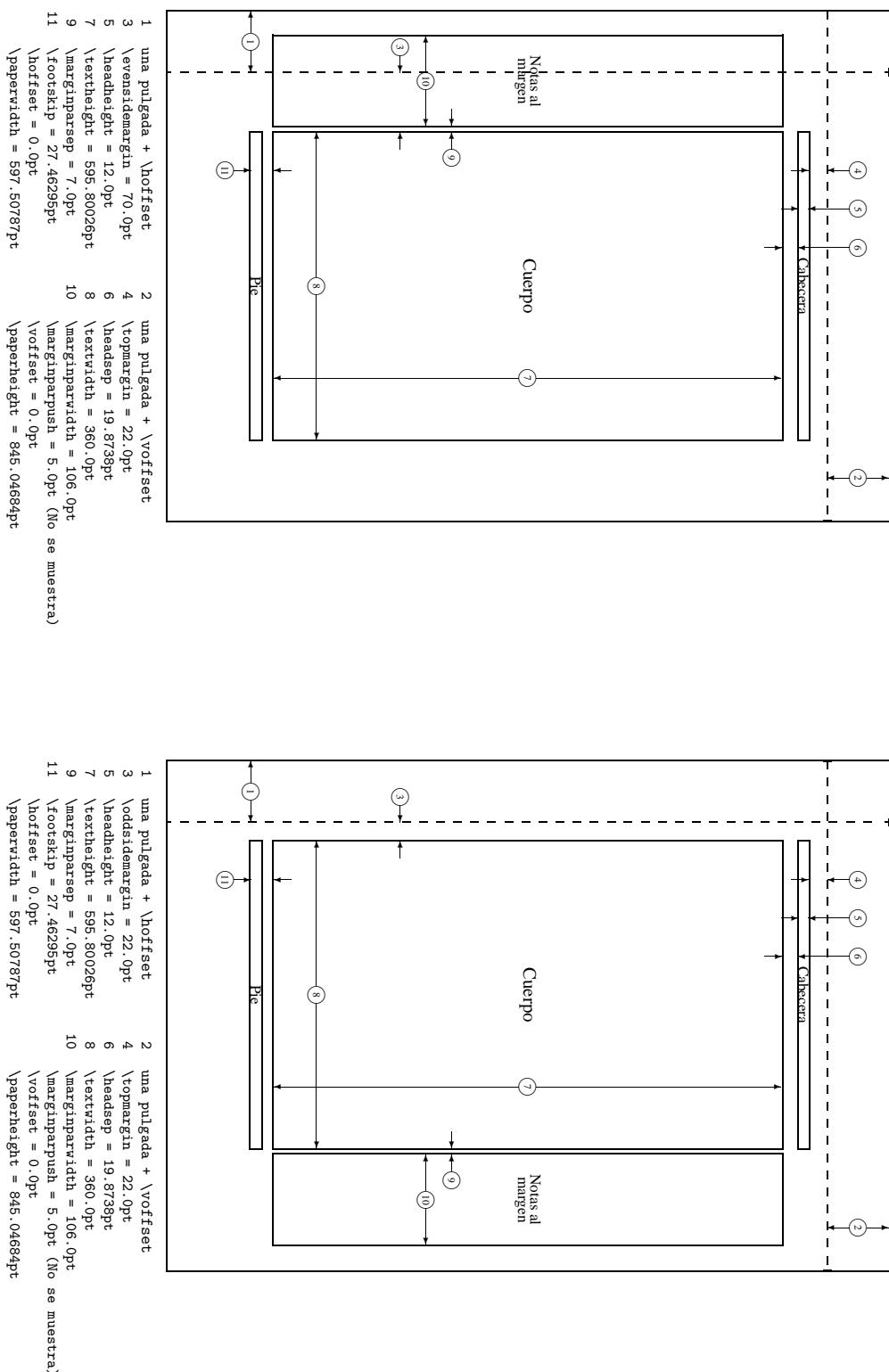


Figura 7.1: Esquema del diseño de las páginas a izquierda y derecha y los parámetros que lo definen, bajo la clase de documento `book`

Hemos de insistir en que L^AT_EX no se ocupa de cambiar otros parámetros cuando se produce un cambio en `\hoffset`; por tanto, es necesario tomar precauciones para que, por ejemplo, si se aumenta `\hoffset`, el texto no sobrepase el borde derecho del papel o para que el texto quede centrado en el papel, tal y como hemos explicado en nuestras consideraciones anteriores.

`\oddsidemargin` Cuando se está utilizando la opción `twoside`, esta longitud se refiere a las páginas a derecha (impares) e indica la distancia que separa el «borde izquierdo de impresión» (véase el apartado sobre `\hoffset`) del texto (véase también `\evensidemargin`). Obsérvese que esta longitud se refiere, por tanto, al margen interior de las páginas cuando el documento es encuadrado. Si no se utiliza la opción `twoside` esta longitud se aplica a todas las páginas.

`\textheight` Esta longitud es la altura normal del cuerpo de la página escrita (excluyendo la cabecera y el pie, pero incluyendo las notas a pie de página). Es conveniente que sea modificada, si se desea, en el preámbulo.

`\textwidth` Esta longitud es la anchura normal de la página escrita, sin incluir el espacio destinado a las notas al margen. Como para la anterior longitud, las posibles modificaciones deben figurar en el preámbulo.

`\voffset` Define el margen vertical de salida o «borde superior de impresión». Los mismos comentarios que en el caso de `\hoffset` se aplican aquí: `\voffset` puede ser definido como positivo o negativo, lo que significa, respectivamente, un incremento o disminución de la distancia vertical que, por defecto, el «visor» o programa de impresión toma desde el borde superior del papel hasta la zona destinada a la impresión.

§7.3 Herramientas de paginación final

CUANDO un documento se considera terminado, somos nosotros, los autores (editores, impresores), los responsables del aspecto final del mismo. Hasta aquí hemos aprendido, y experimentado, que L^AT_EX se ocupa, en la mayoría de los casos, de presentar el documento de acuerdo con criterios profesionales y estéticamente elegantes. No obstante puede ocurrir que en determinadas zonas del documento nos parezca que un espacio horizontal (o vertical) extra mejora el aspecto general del mismo. También puede ocurrir que un determinado corte de página sea inapropiado, porque, por ejemplo, una determinada línea queda sola al acabar o empezar página: son las llamadas líneas *huérfanas* o *viudas*. Si no quiere sufrir este último problema incluya en el preámbulo de su documento el código

<code>\clubpenalty=10000</code>	<code>\widowpenalty=10000</code>
---------------------------------	----------------------------------

que instruyen a L^AT_EX para utilizar márgenes de tolerancia muy exigentes que impiden que una página acabe en una línea huérfana o empiece con una línea viuda, respectivamente.

Para solventar problemas que aparecen en la paginación final de nuestro documento necesitamos comandos que permitan insertar espacios adicionales y comandos que produzcan saltos en una página concreta o que la alarguen.

Si hablamos de espacios verticales, dejar una línea en blanco produce el comienzo de un nuevo párrafo que estará separado del párrafo anterior en sentido vertical una cierta distancia. Si queremos conseguir que esta distancia sea mayor es inútil dejar varias líneas en blanco, pues si lo hacemos L^AT_EX produce el mismo espacio que al dejar una sola línea en blanco. Para dejar un espacio vertical u horizontal, de acuerdo con nuestras necesidades, L^AT_EX dispone de los comandos siguientes:

<code>\vspace{Longitud}</code>	<code>\vspace*{Longitud}</code>
<code>\hspace{Longitud}</code>	<code>\hspace*{Longitud}</code>

que producen un avance vertical o, respectivamente, horizontal de la *Longitud* que se ha indicado. Tal longitud puede ser positiva o negativa y puede indicarse en diferentes unidades de medida, por ejemplo, `\vspace{3cm}`, `\hspace{-15mm}` (véase la lección 17 donde se comentan las longitudes y las unidades de medida admisibles). En el comienzo de una página el comando `\vspace{Longitud}`, si lo hubiere, es ignorado y si se introduce en mitad de un párrafo, espera a que se complete la línea en curso antes de actuar. El comando `\hspace` actúa de forma inmediata, exactamente donde L^AT_EX lo encuentra, salvo que coincida con el principio de una línea, en cuyo caso es ignorado. Utilizando la versión con * se consigue que los espacios en estas situaciones especiales no sean ignorados.

Pueden producirse separaciones verticales predefinidas mediante los comandos:

<code>\smallskip</code>	<code>\medskip</code>	<code>\bigskip</code>
-------------------------	-----------------------	-----------------------

El último de ellos corresponde, aproximadamente, al efecto que produciría dejar una línea en blanco; el anterior es la mitad de dicha longitud vertical, y el primero de ellos, la cuarta parte. Utilizar estas separaciones verticales predefinidas puede tener la ventaja de estar seguros de que siempre utilizamos el mismo espacio extra (por ejemplo, al separar la leyenda de figuras o tablas del texto que le sigue).

Para la separación horizontal los siguientes comandos proporcionan espacios predefinidos:

<code>\quad</code>	espacio de longitud un em
<code>\quad\quad</code>	espacio de longitud dos em
<code>\enskip</code>	espacio de longitud medio em
<code>\,</code>	espacio entre palabras

El primero de ellos corresponde a una longitud un poco mayor que la anchura de la letra M para la fuente en uso (em es una unidad de longitud relativa, véase la lección 17). El último de ellos (que representa el carácter \ seguido de un espacio) corresponde al espacio de separación «normal» entre dos palabras. Este comando ya ha sido comentado en la lección 3.

Los siguientes comandos proporcionan espacios que pueden ser tanto horizontales, por ejemplo en un texto normal, como verticales, por ejemplo entre dos párrafos:

<code>\enspace</code>	espacio de longitud medio em
<code>\thinspace</code>	espacio de longitud 0,16667 em
<code>\negthinspace</code>	espacio negativo de longitud 0,16667 em

Con la misma sencillez con la que los comandos descritos anteriormente se ocupan de la inserción de espacios, el comando

\newpage

se usa para iniciar una nueva página. Otros comandos que producen un salto de página son:

\clearpage

\cleardoublepage

El primero de ellos es similar a \newpage, excepción hecha de que si hay «objetos flotantes» que aún no han sido ubicados (véanse las lecciones 9 y 14), los ubica en una o más páginas sin texto alguno e inicia una nueva página. El comando \cleardoublepage es análogo al comando \clearpage, salvo que si se está utilizando la opción *twoside* (véase el apartado 6.6) de la clase en uso, a fin de imprimir por ambos lados del papel, avanza una o dos páginas, según convenga, hasta situarse en una página impar.

En la paginación final puede ocurrir que una determinada página del documento quede poco «llena» (espacios verticales muy grandes) porque, por ejemplo, un párrafo o un título de sección ha «viajado» a la página siguiente. Si esto ocurre así por una longitud vertical no muy grande podemos tratar de hacer volver el elemento «viajero» a la página actual alargando la altura de la misma mediante el comando:

\enlargethispage{*Longitud*}

\enlargethispage*{*Longitud*}

Este comando alarga la página en la que aparece en la cantidad *Longitud*, sin modificar el resto de los parámetros. También puede acortarse la página, para hacer que el último elemento en ella pase a la siguiente; basta para ello introducir una *Longitud* negativa. De hecho, en las imprentas profesionales se utilizan técnicas similares, con el resultado de que alguna página puede ser ligeramente más larga o más corta que las restantes. La versión con asterisco es más fuerte; puede ocurrir que el compilador atienda la petición de modificar la altura de la página con esta segunda versión, pero no lo haga con la primera.

Ejercicios

7.1  El fichero ejercicio7.1.tex es un documento largo que utiliza la clase article. Efectúa las siguientes tareas con él:

1. suprima la numeración en las páginas primera y tercera;
2. experimente con la inserción del comando \maketitle en la primera página del documento y coloque el comando \thispagestyle{empty} antes y después del mismo, para comprobar que la primera página se numera, o no, en función de la ubicación de \thispagestyle{empty};
3. utilice el comando \enlargethispage para alargar la segunda página;
4. modifique la anchura y altura del texto y su posición respecto al borde del papel.

7.2 Tome un texto de media página y deje un espacio vertical de 32 mm al comienzo de la misma para pegar una foto.

- 7.3 Tome un texto de media página y, tras compilarlo, seleccione una palabra que finalice una línea intermedia en un párrafo. Inmediatamente después de esa palabra en el documento fuente añada el comando `\hspace{2cm}` y compile de nuevo. Repita el proceso con `\hspace*{2cm}` en el lugar de `\hspace{2cm}`. ¿Qué observa?
- 7.4 ¿Qué ocurre si en mitad de un párrafo se incluye el comando `\vspace{2cm}`? Seleccione una palabra en mitad de una línea del párrafo e intente conseguir un salto vertical de 2 cm inmediatamente después de dicha palabra.

PARA SABER MÁS

- ▶ Las cabeceras que hemos utilizado en este libro no son las estándar de L^AT_EX. ¿Le gustaría personalizar sus cabeceras y pies? Las herramientas que necesita para esto las tiene en la sección II.2.4.
- ▶ Como puede observar en la figura 7.1, el control sobre el diseño de una página descansa sobre muchos más parámetros de los que hemos comentado en esta lección. Los que hemos explicado aquí son los básicos. La explicación de todos los parámetros que controlan el diseño de una página la tiene en la sección II.2.5.
- ▶ Hay más, mucho más que decir y aprender sobre espacios horizontales y verticales. En la lección 16 encontrará algunos espacios disponibles en el modo matemático. En la lección 18 hablamos más de espacios y en particular de espacios «elásticos». En los apartados II.1.5.2 y II.8.2 de la segunda parte del libro está el resto de lo que usted pudiera necesitar.
- ▶ Para otras herramientas de paginación final puede consultar la sección II.1.1.3.

Lección

8

Referencias cruzadas



OBJETIVOS

- Etiquetar las unidades de estructura y los entornos numerados para poder referenciarlos.
- Convertir las unidades de estructura y las referencias cruzadas en enlaces de hipertexto.
- Crear manualmente enlaces de hipertexto a destinos previamente definidos.

PAQUETES COMENTADOS: color, hyperref

AS referencias en un documento ayudan al lector a complementar su lectura y, mediante ellas, el autor le invita a prestar atención a un concepto, figura o ejemplo numerado que se encuentra en esta o aquella página. Quienquiera que haya escrito un documento medianamente largo, organizado por capítulos y secciones, que contenga tablas y ejemplos numerados, etc., sabe que, con bastante probabilidad, más de una de estas partes terminará siendo cambiada de sitio o, quizás, introducirá un nuevo ejemplo, que le obligará a cambiar la numeración de todos los que lo siguen. Modificar la numeración de las distintas partes de un documento no representa gran problema, aunque puede ser tedioso; una vez hechas las modificaciones, actualizar las referencias para la nueva numeración puede ser bastante engorroso y, a veces, casi imposible. L^AT_EX simplifica enormemente la realización de estas tareas. Además, al producir un documento PostScript o PDF, nuestras referencias usuales también pueden convertirse fácilmente en «hiperenlaces», como los que estamos acostumbrados a utilizar cuando navegamos por Internet.

§8.1 Las referencias básicas de L^AT_EX

C UANDO utilizamos L^AT_EX podemos referenciar cualquier objeto numerado, siempre que haya sido adecuadamente etiquetado. Así, para poder efectuar una referencia necesitamos dos instrucciones: una para dar nombre a la sección, apartado, figura, etc., a la cual queremos referirnos, y otra para realizar la referencia. L^AT_EX puede hacer este trabajo de forma automática mediante los siguientes comandos:

\label{Etiqueta}	\ref{Etiqueta}
------------------	----------------

donde *Etiqueta* es una cadena de caracteres que puede contener letras, dígitos o signos de puntuación; mayúsculas y minúsculas son diferentes dentro de una *Etiqueta*. Etiquetas válidas, y distintas, son, por ejemplo, Def1, def1 y DEF1.

La utilización del comando \label{Etiqueta} hace que *Etiqueta* guarde la numeración de la unidad de estructura o entorno numerado en el que aparece y que sea el último iniciado, y no terminado, antes del comando \label. Así, por ejemplo, si \label{Etiqueta} se encuentra dentro de una sección (iniciada con el comando \section) que a su vez está dentro de una unidad de estructura iniciada con \chapter, el valor que toma *Etiqueta* es el correspondiente al contador de la unidad \section.

Además de las unidades de estructura existen otros elementos numerados, que veremos en las próximas lecciones, como son los entornos figure (v. §9.4), table (v. §14.4), equation (v. §16.1) y cada uno de los apartados de un entorno enumerate (v. §11.1). La lista de las unidades de estructura numeradas por L^AT_EX se encuentra en la lección 6 (página 50). En la segunda parte del libro se presentan muchos otros entornos y objetos numerados.

El comando \ref{Etiqueta} imprime, en el lugar del documento donde aparece, el número de la unidad de estructura o entorno guardado en *Etiqueta*.

El comando

\pageref{Etiqueta}

se utiliza para obtener la página en la que se encuentra \label{Etiqueta}. Por ejemplo, la referencia que hemos hecho unas líneas antes a la lista de las unidades de estructura numeradas ha sido producida con estos comandos.

El ejemplo 8.1 ilustra cómo se utiliza el comando \label para producir etiquetas en nuestro documento y cómo se utilizan los comandos \ref y \pageref para obtener la información que almacenan.

EJEMPLO 8.1

```
\section{Las referencias básicas de LATEX}
\label{referen}

Cuando utilizamos LATEX{} podemos referenciar
cualquier objeto numerado ...

\section{Algunas precauciones que debemos
tomar}\label{precau}
La utilización de los comandos
anteriores es ...

... se encuentra en la sección~\ref{referen}
(página~\pageref{referen}), y algunos consejos
y recomendaciones sobre el etiquetado están
en la sección~\ref{precau}
(página~\pageref{precau}).
```

8.1 Las referencias básicas de L^AT_EX

Cuando utilizamos L^AT_EX podemos referenciar cualquier objeto numerado ...

8.2 Algunas precauciones que debemos tomar

La utilización de los comandos anteriores es muy simple ...

La lista de las unidades de estructura numeradas por L^AT_EX se encuentra en la sección 8.1 (página 71), y algunos consejos y recomendaciones sobre el etiquetado están en la sección 8.2 (página 73).

§8.2 Algunas precauciones que debemos tomar

La utilización de los comandos que manejan las referencias es muy simple, como hemos visto en el apartado anterior. Sin embargo, para un uso adecuado de los mismos se deben tener presentes las siguientes consideraciones:

- Es conveniente poner etiquetas a todas las unidades de estructura, bien incluyendo el comando `\label{Etiqueta}` al final del argumento del comando de estructura, como en


```
\section{Una sección cualquiera}\label{SuNombre}
```

 o bien inmediatamente después de iniciar la unidad de estructura, como en


```
\section{Una sección cualquiera}\label{SuNombre}
```
- Cada *Etiqueta* debe ser única en nuestro documento, puesto que debe determinar de forma unívoca a un objeto. A este respecto no está de más utilizar etiquetas bastante descriptivas, como por ejemplo


```
\label{SeccionReferencias}
\label{EcuacionDeOndas}
\label{Formula.del.IPC}
```
- *Etiqueta* no puede contener ninguno de los siguientes caracteres reservados: \ % { } # (véase el apartado 4.1).
- Es conveniente utilizar el carácter ~ (véase §4.3) junto con los comandos `\ref` y `\pageref`, para que no se dé la circunstancia de que al intentar obtener «... como se ve en la figura 1.15» mediante, por ejemplo,


```
... como se ve en la figura \ref{FigPrincipal}
```

 podamos obtener «... como se ve en la figura» al final de una línea o de una página y «1.15» al principio de la siguiente (donde 1.15 es la unidad correspondiente a la etiqueta `FigPrincipal`). El código correcto sería entonces


```
... como se ve en la figura~\ref{FigPrincipal}
```
- L^AT_EX produce las referencias cruzadas después de dos compilaciones del documento. En la primera compilación se escriben en el fichero auxiliar (de extensión aux) los datos producidos por los comandos `\label` y que son necesarios para poder cruzar las referencias. En la segunda compilación los comandos `\ref` y `\pageref` instruyen a L^AT_EX para obtener de dicho fichero auxiliar el número del objeto numerado y la página correspondiente.

Algunos errores usuales que se cometan, y que el compilador nos recuerda con claros mensajes, son los siguientes:

- *Utilizar la misma etiqueta más de una vez.* Si hubiésemos utilizado `\label{RefCruz}` para etiquetar la presente sección y también el apartado siguiente, el compilador nos hubiera dado el siguiente mensaje de precaución:

LaTeX Warning: Label ‘RefCruz’ multiply defined

- *Hacer referencias a etiquetas no definidas.* En este caso el compilador nos da mensajes como el siguiente:

LaTeX Warning: Reference ‘subsec:numero’ on page 5

undefined on input line 278

que aparecerá cada vez que utilicemos `\ref{subsec:numero}`.

§8.3 Los enlaces de hipertexto en L^AT_EX

TODOS estamos ya acostumbrados a utilizar documentos interactivos PDF y a movernos en Internet. Una de las características principales de las páginas web, escritas en un lenguaje especial llamado HTML, es la posibilidad de desplazarnos de un punto a otro de la página, o incluso a otras páginas. Esto se consigue por medio de los *enlaces de hipertexto*, también llamados *hiper-enlaces*. Con L^AT_EX también es posible crear documentos «navegables», para lo que necesitamos usar el paquete `hyperref`, especialmente concebido para documentos en formato PDF o PostScript, aunque también se puede utilizar si la salida de la compilación es un documento DVI.

El objetivo principal del paquete `hyperref`, desarrollado por S. Rathz y H. Oberdiek [57], es la generación automática de enlaces de hipertexto para todos los tipos de referencias cruzadas que L^AT_EX establece: las entradas en el índice general enlazan con las unidades de estructura (véase §6.2), las citas bibliográficas (véase §15) enlazan con sus entradas en la bibliografía, etc. Por generación automática queremos decir que el paquete extiende la funcionalidad de los comandos de L^AT_EX que llevan a cabo estas tareas, lo que significa que, salvo cargar el paquete en el preámbulo del documento, no es necesaria ninguna acción adicional para obtener cualquier referencia en las dos formas posibles: la tradicional en papel y el enlace de hipertexto en nuestro documento electrónico.

No todos los programas utilizados para visualizar nuestros documentos soportan enlaces de hipertexto, aunque los más importantes sí. El paquete soporta diversos «controladores» u opciones, como era el caso del paquete `color` (véase el apartado 5.5); de hecho ambos paquetes poseen los mismos controladores. Recordemos que las opciones correspondientes a los principales controladores son las siguientes¹:

`dvips` Utiliza la sintaxis apropiada para que el programa DVIPS genere un archivo PostScript con hiperenlaces. La opción `dvips` para `hyperref` puede sustituirse, de forma totalmente equivalente, por `nativepdf` o `pdfmark`.

`pdftex` Para ser utilizado con el programa PDFL^AT_EX, que genera directamente un archivo PDF.

Dado que `hyperref` redefine muchos comandos de L^AT_EX, es conveniente que el paquete sea el último en ser cargado, para garantizar así su correcto funcionamiento.

El paquete soporta una gran cantidad de opciones que se indican como una lista de parámetros a los que se asignan los valores correspondientes; más concretamente, la sintaxis es:

Parám₁=Valor₁, Parám₂=Valor₂, Parám₃=Valor₃, ..., Parám_N=Valor_N

¹En algunas instalaciones no es siempre necesario incluir una opción para el controlador (véase la página 45).

Algunos parámetros son de tipo lógico o «booleano», es decir, sólo admiten los valores `true` (verdadero) o `false` (falso), en cuyo caso, si se les quiere dar el valor `true` bastará incluir el nombre del parámetro, sin necesidad de la parte `=true`.

Para especificar un valor para las distintas opciones podemos utilizar también el comando

```
\hypersetup
```

Por ejemplo, una típica línea de uso de este comando es la siguiente:

```
\hypersetup{colorlinks=true,linkcolor=Blue}
```

A continuación se presentan dos de las opciones más importantes del paquete; entre paréntesis figura, cuando procede, el valor por defecto.

`draft` Todas las opciones de hipertexto son desactivadas. Esta opción también puede aparecer en la clase de documento o como opción al cargar el paquete (v. pág. 58).

`colorlinks` (`false`) El paquete permite la utilización de colores en la construcción de los hiperenlaces, para lo cual debemos activar la opción `colorlinks`, incluyéndola dentro del comando `\hypersetup`. Los colores utilizados dependen del tipo de enlace, y se distinguen básicamente cinco tipos. Más detalles en el apartado PARA SABER MÁS.

§8.4 Comandos para crear enlaces

ADEMÁS de convertir en hiperenlaces las entradas en los índices y las referencias, el paquete `hyperref` proporciona comandos específicos para crear manualmente nuevos enlaces. Como ocurría con las referencias ordinarias, para poder realizar un hiperenlace necesitamos dos instrucciones: una para etiquetar la parte del documento con la que queremos enlazar y otra para realizar el hiperenlace propiamente dicho.

Para realizar estas tareas, el paquete `hyperref` proporciona los siguientes comandos:

```
\hypertarget{Etiqueta}{Objeto1}
```

```
\hyperlink{Etiqueta}{Objeto2}
```

El comando `\hyperlink` convierte el argumento `Objeto2` en un hiperenlace para conectar con el punto de destino `Objeto1`, denominado `Etiqueta` y que ha sido previamente definido (o será posteriormente definido) con el comando `\hypertarget`.

EJEMPLO 8.2

```
Comienzo de la
\hypertarget{link1}{primera página}.
\clearpage
```

```
Una página intermedia. \clearpage
```

```
Última página con enlace a la primera
\hyperlink{link1}{justo aquí}
```

```
Comienzo de la primera página.
```

```
.....
```

```
Uma página intermedia.
```

```
.....
```

```
Última página con enlace a la primera justo aquí
```

En la primera página se ha definido el hiperenlace `link1` en las palabras «primera página». Un enlace en la tercera página, en las palabras «justo aquí», conecta con el hiperenlace `link1` definido previamente. Si la opción `colorlinks` estuviera activada, las palabras «justo aquí» aparecerían coloreadas en lugar de encerradas en un marco

Si vamos a definir muchos hiperenlaces en nuestro documento, entonces una buena idea puede ser agruparlos por categorías. Para ello disponemos del comando

```
\hyperdef{Categoría}{Nombre}{Objeto}
```

que asigna a *Objeto* la etiqueta *Categoría.Nombre*, definiéndolo así como punto de destino de futuros enlaces \hyperlink. Es equivalente a \hypertarget{Categoría.Nombre}{Objeto}.

Un ejemplo de las categorías de hiperenlaces es el siguiente. Las páginas de los documentos PDF creados con PDFLATEX están etiquetadas: la etiqueta de cada página es page.N, donde N indica el número de página. De modo que un enlace, por ejemplo, a la página 235 se puede construir escribiendo \hyperlink{page.235}{Texto}. Alternativamente, para enlazar con una página podemos utilizar, recuerde, sólo con PDFLATEX, el siguiente comando:

```
\hyperpage{N}
```

donde N indica el número de la página; el comando es equivalente a \hyperlink{page.N}{N}.

En el comando anterior es necesario conocer el número de la página con la cual queremos enlazar, lo cual no suele ser habitual. La manera más natural de utilizar los dos comandos anteriores es en combinación con el comando \pageref, que nos proporciona el número de la página en la que aparece determinada etiqueta. Por ejemplo, para construir un enlace a la página donde aparece una figura (v. §9.4), primero etiquetamos la figura, mediante el comando \label, y después escribimos, en el lugar donde queramos insertar el hiperenlace, \hyperlink{page.\pageref{EtiFigura}}{Texto}, suponiendo que *EtiFigura* es el nombre que hemos utilizado para la etiqueta.

Cuando trabajamos con documentos PDF, es habitual enlazar unos archivos PDF con otros, de manera que vamos navegando no sólo dentro de un documento PDF, sino también entre documentos diferentes. Para crear un hiperenlace a otro documento PDF podemos utilizar el siguiente comando:

```
\href{file:Archivo}{Objeto}
```

que convierte el argumento *Objeto* en un hiperenlace que conecta con el nuevo documento PDF de nombre *Archivo*.

Ejercicios

- 8.1 En un documento producido con la clase article cree dos secciones. Etiquete las dos secciones y en cada sección realice una referencia a la otra sección. Observe que se pueden referenciar objetos tanto anteriores como posteriores en el documento fuente.
- 8.2 Compile el documento fuente ejercicio8.2.tex con PDFLATEX; observará que el fichero PDF resultante (ejercicio8.2.pdf) no contiene hiperenlaces. Realice los cambios oportunos en el documento fuente para obtener el documento ejercicio8.2_sol.pdf, en el cual se han activado los hiperenlaces. La solución puede encontrarla en el archivo de nombre ejercicio8.2_sol.tex.

PARA SABER MÁS

- ▶ Las referencias básicas de L^AT_EX se limitan al propio documento. Sin embargo, en ocasiones puede ser necesario realizar referencias a las unidades de estructura o entornos numerados de otro documento. Sobre este tema podrá encontrar más información en el apartado PARA SABER MÁS del capítulo 2.
- ▶ El paquete `hyperref` dispone de muchas más opciones y permite configurar totalmente la apariencia de los hiperenlaces, en particular su color según el tipo. Algunas de estas opciones son específicas para los documentos en formato PDF. En la sección II.6.1 encontrará el listado completo de opciones del paquete `hyperref`.
- ▶ Además de los comandos para creación de enlaces que hemos visto en la lección, el paquete `hyperref` dispone de otros comandos útiles para la creación de enlaces a direcciones de Internet. En la página 403 se explican con detalle.
- ▶ El comando `\href` no sólo permite enlazar con otros documentos PDF, si no que puede enlazar con cualquier otro tipo de documentos, así como con otros recursos que tengamos a nuestra disposición (bien en nuestro computador, bien en Internet). Consulte la sección II.6.1 para más detalles.

Inclusión de gráficos



OBJETIVOS

- Conocer las principales dificultades a las que un usuario se enfrenta a la hora de incluir un gráfico en un documento L^AT_EX.
- Incluir un gráfico en un documento L^AT_EX, con algunas opciones básicas.
- Ubicar un gráfico como una figura: con numeración y leyenda explicativa.
- Generar la lista de todas las figuras de un documento.

PAQUETES COMENTADOS: graphicx

CON frecuencia se ha mantenido, entre los conocedores de L^AT_EX, la idea de que la mayor debilidad de este sistema reside en la inclusión de gráficos en los documentos. Probablemente es cierto y, casi seguro, lo era hasta hace unos años. La situación ha mejorado enormemente, no sólo por las mejoras específicas introducidas en L^AT_EX, sino también por el desarrollo de herramientas externas, de libre distribución, que permiten «tratar» un gráfico para que pueda ser incorporado a un documento L^AT_EX.

La razón principal de esta debilidad, sea ésta importante o no, se remonta al momento de la creación de T_EX. En aquellos años, los setenta, la presencia de instrumentos gráficos en los computadores era prácticamente nula, no se habían desarrollado formatos electrónicos capaces de almacenar información gráfica con un tamaño razonable y, por supuesto, no existía un formato estándar de datos para los gráficos que, aún hoy, todavía no ha llegado. A pesar de ello, Donald Knuth, con asombrosa clarividencia, dejó abierta la posibilidad de que T_EX fuera capaz, en el futuro, de incorporar gráficos y otras mejoras tecnológicas (como los hiperenlaces, que ya hemos visto funcionar en la lección 8). Poco importa, en este lugar, saber cómo lo consiguió; bastará, por ahora, convencerse de que es muy fácil incorporar gráficos a nuestros documentos L^AT_EX, siempre que estos gráficos estén en un formato adecuado. Como el lector descubrirá a lo largo de la lección, los «formatos adecuados» cubren un amplio espectro del conjunto de los formatos gráficos habituales.

Debemos tener en cuenta, eso sí, que en materia de incorporación de gráficos perderemos una parte del carácter universal, independiente de la plataforma, de L^AT_EX. A pesar de todo ello el nivel actual de estandarización de L^AT_EX permite reducir enormemente esta pérdida.

§9.1 Distintos tipos de gráficos

La variedad de gráficos que pueden ser incluidos en nuestros documentos es enorme. Intentar una clasificación de los mismos es tarea difícil y ardua, en parte porque pueden ser clasificados en base a distintos conceptos: su origen, su contenido, su formato de datos (en el caso de los gráficos electrónicos), etc. Con el objetivo de comprender los problemas que afrontamos a la hora de incluir un gráfico en un documento \LaTeX , podemos realizar una primera clasificación, simple, que atiende exclusivamente al método de generación del formato electrónico del gráfico:

- Gráficos externos a \LaTeX .
- Gráficos generados con \LaTeX o con alguna herramienta capaz de generar una salida en lenguaje comprensible por \LaTeX , ya sea \LaTeX básico, extensiones proporcionadas por algún paquete o directamente el visor de los documentos compilados.

La diferencia esencial entre estos dos tipos de gráficos es que los primeros constituyen un «objeto cerrado» para \LaTeX , lo que no ocurre en el segundo caso. Este carácter cerrado tiene enormes consecuencias: la integración del gráfico con el resto del documento se verá irremediablemente comprometida. Por ejemplo, los tipos (fuentes) del texto y del gráfico no coincidirán, una situación reprobable si se busca un documento de calidad. Sólo si el programa con el que se ha generado el gráfico es capaz de manejar los mismos tipos que \LaTeX , algo improbable, se podrá eliminar este problema. De forma más general, este carácter cerrado del objeto gráfico puede impedir dotar de un estilo propio a los gráficos, adaptado, de alguna manera, al estilo general del documento.

Naturalmente los problemas anteriores se ven reducidos si el gráfico es, por ejemplo, una fotografía: en este tipo de gráfico es menos natural pensar en la inclusión de texto (aunque el problema de «estilo general» subsiste). Este aspecto sugiere una clasificación que atienda al contenido del gráfico, que podría darse en los términos que figuran a continuación.

- Gráficos artísticos. Aquellos que pueden contener todo tipo de elementos complicados. Se denominan, en ocasiones, *de trama*, ya que su forma de reproducción se basaba en una trama de puntos o líneas. Son aquellos gráficos que en la imprenta clásica eran creados por el artista (grabador) y no por el cajista, de ahí que podamos denominarlos bajo el nombre genérico de «trabajos artísticos». Algunos ejemplos serían: dibujos a mano alzada o técnicos realizados manualmente, fotografías de cualquier tipo, cualquier tipo de diseño gráfico. Todos ellos serían generados en un formato electrónico ya sea mediante un escáner, una cámara digital o cualquier herramienta gráfica para nuestro computador.
- Gráficos lineales. Aquellos que contienen únicamente elementos sencillos: puntos, líneas rectas o curvas, regiones cerradas, eventualmente coloreadas, etc. Suelen estar desglosados en varios «objetos» o componentes gráficos independientes, cada uno de los cuales es susceptible de ser reutilizado en el propio gráfico, de forma idéntica a sí mismo o mediando un cierto escalado o rotación, para ir formando objetos más y más complejos. A menudo esta orientación hacia los objetos permite que posean una estructura interna, lo que, a su vez, facilita que puedan ser descritos por un lenguaje abstracto. Algunos ejemplos podrían ser: curvas planas o espaciales (con o sin ejes de coordenadas), diagramas conmutativos o de flujo, circuitos electrónicos, esquemas arborescentes, diagramas de barras, etc.

Podemos observar un cierto paralelismo entre las respectivas categorías de ambas clasificaciones: el «trabajo artístico» se suele generar con herramientas externas a L^AT_EX (parece evidente que L^AT_EX no es un medio adecuado al diseño gráfico general). La segunda de las categorías (lineal) se presta mucho más a un lenguaje formal, como el de L^AT_EX. Se trata, en todo caso, de un cierto paralelismo, en absoluto de una identificación.

En esta lección sólo abordaremos la forma de incluir gráficos externos. La generación de gráficos desde L^AT_EX o paquetes complementarios será tratada en la segunda parte de este libro.

§9.2 El paquete `graphicx` y los formatos gráficos incorporables

CENTRÁNDONOS en los gráficos externos, de «tipo cerrado», podemos preguntarnos: ¿cuál es la vía mediante la cual podemos incorporar gráficos a nuestros documentos L^AT_EX?, ¿qué tipos de gráficos podemos incorporar? El nivel de estandarización alcanzado actualmente por L^AT_EX permite simplificar al máximo la inclusión de gráficos. Esta simplificación se alcanza por medio de un paquete: el paquete `graphicx`.

Este paquete debe ser cargado con una opción que indica el programa de visionado o impresor (denominado a menudo controlador) que vamos a utilizar, y el paquete, según la opción indicada, transformará una sentencia universal (que aprenderemos en el apartado siguiente) en el lenguaje adecuado para dicho programa. El documento obtenido de esta forma no es independiente de la plataforma, ni siquiera de la herramienta de visualización, pero adaptarlo a otro entorno sólo supone, en principio, cambiar el nombre de una opción: la opción de `graphicx`.

Así pues, lo primero para incluir un gráfico es cargar el citado paquete con el controlador adecuado; los controladores disponibles son los mismos que para los paquetes `color` e `hyperref`. Como ya se señaló, en los apartados 5.5 y 8.3, las dos opciones de controlador fundamentales son `dvips` y `pdftex`¹. En esta lección comentamos brevemente algunos aspectos de un nuevo controlador, especificado con la opción `dvipdfm`.

La precaución anterior sería suficiente para poder incluir gráficos en un documento L^AT_EX, si no fuera porque los formatos gráficos (electrónicos) que podemos incorporar dependen del controlador elegido; es decir, cada controlador es capaz de entender y gestionar sólo algunos formatos gráficos. Por tanto, debemos ser conscientes de la siguiente restricción, consecuencia del hecho anterior: *cada formato de salida admite únicamente algunos formatos de gráficos electrónicos*. Así pues, hemos de tomar algunas decisiones:

- o bien elegimos un controlador que sea capaz de gestionar el formato de los gráficos que ya tenemos (controlador para una herramienta de visualización que, naturalmente, debemos tener instalada en nuestro sistema),
- o bien convertimos nuestros gráficos a un formato comprensible por nuestro visor (controlador) preferido.

¹Cuando, en la compilación, se carga el paquete `graphicx` se lee el fichero `graphics.cfg`, un archivo de configuración. Algunos compiladores, como por ejemplo MiK^AT_EX, distribuyen este fichero con un contenido tal que, en algunos casos, permite evitar la opción *Controlador* del paquete; concretamente, se carga automáticamente la opción `dvips`, salvo que se esté compilando con PDFL^AT_EX, en cuyo caso la opción cargada es `pdftex`, véase la página 45.

Quizás sea el formato de salida (resultado final) el que nos interese determinar con antelación: ¿necesitamos un documento en formato DVI, PostScript o PDF? Si el destino final es un documento impreso en papel, y que no requiera una calidad profesional, cualquiera de los formatos podría ser bueno, y, en ese caso, el formato de los gráficos puede ser determinante. Pero si, por ejemplo, deseamos un documento para ser publicado en Internet, entonces, seguramente, la buena elección es PDF, y habremos de adaptar nuestros gráficos para obtener esta salida.

Puesto que hay una gran cantidad de controladores no vamos a precisar aquí los formatos que cada uno de ellos es capaz de gestionar. Nos limitaremos a algunos controladores básicos que ya fueron presentados en la lección 1 (§1.2 y §1.4).

DVI Hay numerosos visores de ficheros DVI: YAP, DVIWIN, DVIWINDO, etc., en MS-WINDOWS, y XDVI, KDVI en LINUX/UNIX. Cada uno de ellos sigue una estrategia diferente para la interpretación de formatos gráficos. Por ejemplo, YAP, distribuido junto con MiK_TE_X, utiliza distintos filtros y programas de conversión que se activan en el fichero de configuración de MiK_TE_X; el visor DVIWIN, por el contrario, utiliza los filtros nativos de MS-WINDOWS. Es posible, de esta forma, incluir varios formatos: PCX, WMF, etc.

En lugar de describir aquí todos los detalles sobre estos procesos y su configuración, nos contentaremos con citar el caso de los ficheros gráficos de formato BMP (gráficos de tipo «bitmap»). Estos ficheros son aceptables por una buena parte de los visores de archivos DVI (al menos los de MS-WINDOWS). Tienen, sin embargo, un problema: los ficheros BMP no contienen indicación de su tamaño, por lo que, a la hora de incluirlos en un documento L_AT_EX, con el comando descrito en el apartado siguiente, habrá que indicar siempre la altura y la anchura del gráfico (ya sea la que posee naturalmente u otra, si es que queremos cambiar su escala). Al citado problema habría que añadir el inconveniente adicional que supone el enorme tamaño electrónico que poseen los archivos gráficos con este formato.

DVIPS Los formatos gráficos admitidos son: PostScript (PS), PostScript encapsulado (EPS), PCX (versión 0) y BMP, este último con el mismo problema sobre el tamaño antes citado y sólo en blanco y negro.

DVIPDFM Este ejecutable gestiona los siguientes formatos gráficos: JPG, JPEG, PNG y PDF. Para todos estos formatos el compilador (con este controlador) busca un fichero que indique las dimensiones del gráfico (con extensión bb). En la distribución de DVIPDFM se incluye un ejecutable (EBB) que genera, para los tipos anteriores de ficheros, el correspondiente fichero .bb con las dimensiones naturales. Además DVIPDFM puede ser configurado para aceptar documentos dvi que contengan gráficos PostScript, incluidos con la opción dvips de graphicx; esto significa que al ejecutar DVIPDFM sobre tales ficheros, el programa ejecuta interna y automáticamente un convertidor de los gráficos a formato PDF (uno de estos posibles convertidores es, precisamente, GHOSTSCRIPT, véase §1.4).

PDFL_AT_EX Los gráficos de formato JPG, JPEG, TIF, TIFF, PNG y PDF son aceptados por este compilador y ninguno de ellos plantea problema alguno.

Cuando nos dispongamos a incluir un gráfico en un documento L_AT_EX procederemos, pues, cuidadosamente: si el formato electrónico del gráfico que poseemos es uno de los aceptados por el controlador (o formato de salida final) que deseamos, entonces todo va bien. En caso contrario

necesitaremos o bien optar por otro formato de salida o bien convertir el formato electrónico del gráfico. Si necesitamos incluir más de un gráfico, entonces las cosas se complican: puede que unos gráficos sean buenos y otros no... Consulte el apartado PARA SABER MÁS para localizar algunas estrategias para hacer frente a estas dificultades.

§9.3 El comando para la inclusión de gráficos

Hemos afirmado ya que, para la inclusión de gráficos externos, el paquete `graphicx` procede traduciendo internamente un «comando universal» al lenguaje adecuado al controlador declarado como opción. Este «comando universal» existe en dos versiones; su sintaxis general es:

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

donde *Archivo* es el nombre del archivo gráfico que deseamos incluir y *ListaOpciones* consiste en una lista de algunas de las opciones disponibles, con una sintaxis determinada, que detallamos un poco más adelante.

En general *Archivo* debe incluir el nombre completo: el camino² o «path» y la extensión, con algunas salvedades: 1) no es necesario incluir el camino si el *Archivo* se encuentra en la misma carpeta que el archivo que se está compilando o si se encuentra en una de las carpetas en las que `LATEX` busca archivos³; 2) no es necesario incluir la extensión si el *Archivo* posee la extensión que el controlador elegido espera por defecto (por ejemplo, con la opción `dvips` se espera un archivo de extensión `eps`, mientras que con la opción `pdftex` se espera un archivo PDF⁴).

Las dos versiones de `\includegraphics`, con y sin asterisco final, se diferencian en el tratamiento del gráfico, una vez especificado su tamaño; la diferencia concreta se clarifica en el apartado sobre la opción `clip`, que figura a continuación.

Opciones básicas de `\includegraphics`

El argumento *ListaOpciones* de `\includegraphics` consiste en una lista de parámetros a cada uno de los cuales se asigna un valor; más concretamente, la sintaxis es la indicada en la página 74, a propósito del comando `\hypersetup`.

A continuación figura una lista de los parámetros más básicos para `\includegraphics`.

²En algunos sistemas operativos, como MS-WINDOWS, el signo para indicar los directorios en el camino de un archivo es la antibarra (\); en estos sistemas se debe sustituir dicho signo, en el documento fuente, por la barra (/), pues, en caso contrario, el compilador tomaría el código por el nombre de un comando.

³Estas carpetas de búsqueda dependen de la instalación del compilador; a menudo suelen estar definidas por una variable global del sistema cuyo nombre puede ser `TEXINPUT` o similar.

⁴Con algunos compiladores, por ejemplo con MiK_ET_EX, es posible automatizar el proceso, gracias al contenido del archivo de configuración de `graphicx` (véase la nota 1): bastará disponer de los dos formatos, EPS y PDF (o JPG), de cada gráfico, incluir cada *Archivo* sin extensión (en el argumento de `\includegraphics`) y no especificar controlador al cargar el paquete `graphicx`.

width Con este parámetro se predetermina la anchura que se dará al gráfico insertado, escalándolo si fuese necesario. Si el controlador elegido es capaz de leer la anchura del gráfico, ya sea en el propio *Archivo* o en alguno auxiliar, entonces este parámetro puede no ser incluido, en cuyo caso el gráfico se insertará con su anchura natural.

height Es un parámetro enteramente análogo al anterior, pero respecto a la altura del gráfico. Si se incluyen los dos parámetros **width** y **height** y éstos no están en la misma proporción que las dimensiones naturales del gráfico, éste quedará deformado (a no ser que se incluya el parámetro **keepaspectratio**). Si se incluye únicamente uno de los dos, el gráfico será escalado, sin deformaciones, hasta alcanzar la dimensión especificada (siempre que el controlador pueda leer las dimensiones naturales).

keepaspectratio Es un parámetro de tipo lógico. Si se le asigna el valor **true** el gráfico será escalado, sin producir ninguna distorsión, para que no exceda ni de la anchura ni de la altura determinadas por las opciones **width** y **height**.

scale Determina un factor de escala que se aplicará en ambas direcciones.

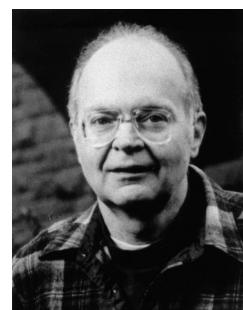
clip Es un parámetro de tipo lógico. Si su valor es **true**, el gráfico será recortado a las dimensiones de la caja especificada por los demás parámetros. En caso contrario, parte del gráfico sobrepasará dicha caja y puede superponerse con el texto exterior al gráfico. La versión con estrella, **\includegraphics***, es equivalente a **\includegraphics[clip]**.

draft Es un parámetro de tipo lógico. Si su valor es **true**, entonces el archivo gráfico *Archivo* no será incluido; en su lugar aparecerá el nombre de *Archivo* y se reservará el espacio correspondiente a la caja que determinen los restantes parámetros. El objetivo de este parámetro es acelerar la compilación y la visualización del documento, y es útil cuando debemos compilar muy a menudo; naturalmente, para la versión final habrá que eliminar dicha opción (o asignarle el valor **false**). Esta opción existe también como opción general de **graphicx** (recuérdese el comentario sobre opciones de la clase de documento en la página 60).

EJEMPLO 9.1

```
\usepackage[dvips]{graphicx}
\begin{document}
...
He aquí la imagen de interés:
\centerline{%
\includegraphics[width=3.5cm,clip]{DKnuth.eps}}
}
¿Adivina de quién se trata?
```

He aquí la imagen de interés:



¿Adivina de quién se trata?

Inclusión de un fichero gráfico con formato PostScript encapsulado; el documento se compila con **LATEX** y se ejecuta posteriormente el programa **DVIPS** (véase el esquema de funcionamiento en la página 6)

EJEMPLO 9.2

```
\usepackage[pdftex]{graphicx}
\begin{document}
...
He aquí la imagen de interés.
¿Adivina de quién se trata?\[3pt]
\centerline{%
\includegraphics[height=3cm]{DKnuth.jpg}
}}
```

He aquí la imagen de interés. ¿Adivina de quién se trata?



Inclusión de un fichero gráfico con formato JPG; el documento se compila con PDFLATEX

§9.4 Las figuras como objetos flotantes

HASTA aquí hemos explicado cómo hacer para poder insertar algunos gráficos en un documento LATEX. En este apartado nos planteamos un problema distinto, que se refiere a la composición del texto en general; el problema es cómo y dónde ubicar los gráficos. En general, en este apartado, nos referiremos a las «figuras» en lugar de los «gráficos». La razón de este cambio de nomenclatura es sencilla: cualquier elemento de tipo gráfico, complementario al texto y que suele numerarse para poder ser referenciado, será considerado una figura, proceda de un archivo gráfico o no.

Este tipo de elementos suele plantear problemas serios en la composición de un documento: se trata de elementos a menudo de un tamaño considerable, y ninguno de ellos puede ser partido entre dos páginas (como un párrafo es partido en líneas), por lo que deben ser incorporados completamente en una página; ¿qué ocurre entonces si no cabe en la página que se está escribiendo? Un problema adicional proviene de la conveniencia de que se traten como elementos referenciables: deben situarse lo más cerca posible de la primera llamada a ellos. Todos estos problemas hacen aconsejable, y casi imprescindible, convertirlos en *elementos flotantes*: objetos que seguramente no aparecerán, en el documento final, en el mismo lugar relativo en el que se incluyeron en el documento fuente y cuya ubicación definitiva se alterará en función de numerosas circunstancias. Es el compilador TEX el que los hace *flotar*, y esto, lejos de ser un inconveniente, porque el que escribe desconoce cuál será su posición definitiva, es una enorme ventaja: no necesitamos resolver el difícilísimo problema de la ubicación por nosotros mismos. Puede parecer exagerado calificar este problema como «difícilísimo», pero no lo es: bastará pensar en un documento breve con varias figuras.

Hay dos tipos de objetos flotantes predefinidos en LATEX: las figuras, de las que nos ocupamos en este apartado, y los cuadros (o tablas), que serán tratados en el apartado 14.4. La forma de convertir un elemento cualquiera en un objeto flotante de tipo figura es mediante el entorno:

```
\begin{figure}[Posición]
Objeto
\caption[TextoLeyendaÍndice]{TextoLeyenda}
\end{figure}
```

Mediante este entorno `figure` se trata al *Objeto* como flotante de tipo figura. Este *Objeto* puede ser cualquier cosa: uno o varios párrafos de texto, texto escalado, un dibujo lineal, un gráfico

externo (incluido con `\includegraphics`) o cualquier combinación de estos y otros elementos. El *Objeto* no es procesado de ninguna manera adicional, tan sólo se decide el lugar más apropiado para su ubicación; por lo tanto, si, por ejemplo, queremos que contenga algún elemento centrado, dicho *Objeto* debe contener los comandos necesarios para centrar el citado elemento.

La sintaxis del entorno `figure` posee otros elementos que se describen a continuación.

Posición Es un argumento optativo que indica el lugar de la página que se prefiere para ubicar la figura. Puede incluir uno o varios de los valores siguientes: `h`, `t`, `b`, `p`. Cada uno de los valores incluidos se corresponde con una posible ubicación del objeto flotante:

- `h` del inglés «*here*», es decir, aquí: en el lugar del documento fuente en el que se ha escrito el entorno `figure`;
- `t` del inglés «*top*», es decir, en la parte superior de una página normal (una página de texto usual, por contraposición a una página que sólo contiene objetos flotantes, como se indica en el último de los valores);
- `b` del inglés «*bottom*», es decir, en la parte inferior de una página normal;
- `p` en una página que no contiene texto, sino únicamente objetos flotantes.

El orden en el que aparezcan estos valores es irrelevante. L^AT_EX sólo utiliza las posiciones correspondientes a las letras que se especifican; es decir, si, por ejemplo, no aparece el valor `b`, nunca ubicará un entorno `figure` en la parte inferior de una página. Hay, sin embargo, una excepción a esta regla: si `h` es la única especificación, L^AT_EX puede no respetar la orden, ya que las reglas internas que posee no siempre le permiten ubicar «aquí» el *Objeto*.

En caso de no incluir este argumento *Posición* L^AT_EX le asigna, por defecto, un valor que, normalmente, es `tbp` (aunque puede cambiar con la clase de documento).

El argumento *Posición* puede incluir, además, un signo de admiración (!). Si está presente L^AT_EX se vuelve menos cuidadoso o menos exigente; así, por ejemplo, si una figura no es ubicada en el lugar deseado mediante `\begin{figure}[h]`, quizás sí se consiga hacerlo con `\begin{figure}[h!]`. Si L^AT_EX necesita cambiar los valores de *Posición* dados por el usuario, informará de ello (mediante un mensaje de precaución, Warning).

`\caption` Es un comando optativo, dentro del entorno `figure`, y sirve para ponerle al *Objeto* una leyenda descriptiva (también llamada *pie* o *epígrafe*). En caso de que se utilice puede colocarse antes o después del *Objeto*, según el lugar en el que se desee que aparezca dicha leyenda.

Si se utiliza este comando el entorno `figure` correspondiente es numerado. El contador que se encarga de numerar las figuras es el contador `figure` (véase el apartado 17.1). L^AT_EX se ocupa por sí mismo de anteponer al argumento *TextoLeyenda* de `\caption` un antetítulo: la palabra ‘Figura’ (con la opción `spanish` de `babel`), seguida del número de orden que le corresponde, y a continuación dos puntos⁵.

⁵ Aquí L^AT_EX mantiene un error: si se desea que se numere la figura sin incluir ninguna leyenda descriptiva, parece lógico incluir la sentencia `\caption{}`, es decir, `\caption` con argumento vacío; esto es admisible, pero el error reside en que, en ese caso, aparecerá en el pie de la figura algo como ‘Figura 1: ’ (nada sigue al signo de dos puntos), que, como se ve, resulta inaceptable. Una solución a este problema se encuentra en la página 308.

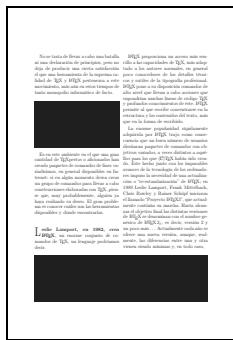


Figura 9.1: Una página, con texto a dos columnas, con dos figuras: un entorno `figure` (en posición h) y un entorno `figure*` (en posición b)

Es muy conveniente, casi diríamos imprescindible, dado el carácter flotante, utilizar el comando `\label` para posibilitar las referencias cruzadas a la figura (v. pág. 72). Para hacerlo correctamente el comando `\label` debe ser incluido dentro del argumento obligatorio del comando `\caption` o inmediatamente después del mismo.

Además el comando `\caption` realiza las tareas necesarias para que la figura aparezca en el índice de figuras (si es que éste es incluido por el usuario). El contenido del argumento obligatorio *TextoLeyenda* será el que, en su caso, aparecerá en el índice de figuras, a menos que se utilice el argumento optativo *TextoLeyendaÍndice*, en cuyo caso será el contenido de éste el que aparezca en dicho índice.

Existe también el entorno `figure*`, con sintaxis idéntica a la de `figure`:

```
\begin{figure*}[Posición]
Objeto
\caption[TextoLeyendaÍndice]{TextoLeyenda}
\end{figure*}
```

La diferencia entre ambos es sencilla: los dos se comportan exactamente igual salvo que se esté escribiendo a dos columnas (con la opción `twocolumn` de la clase de documento, véase §6.6, o con el comando `\twocolumn`, véase §12.1); en ese caso la versión normal coloca las figuras en una columna, mientras que la versión con asterisco las coloca a todo el ancho de la caja o mancha (véase la figura 9.1); esta última opción es pues la apropiada si el *Objeto* incluido en el entorno `figure` posee un ancho mayor que el de las columnas.

Para generar el índice de figuras (naturalmente sólo de aquellas numeradas, es decir, en las que se ha incluido el comando `\caption`), basta incluir el comando

```
\listoffigures
```

Este comando generará el índice, en el lugar del documento en que se haya incluido, después de dos compilaciones: la primera incluye información en un fichero auxiliar (de nombre *el* del documento que se está compilando y de extensión *lof*, por «*list of figures*») y la segunda lee,

y vuelve a escribir, dicha información (es exactamente lo mismo que ocurre con el fichero de extensión `toc` correspondiente al índice general, véase la página 57). En general, se necesitarán dos compilaciones para tener un índice actualizado.

El índice de figuras se inicia con el título genérico «Índice de figuras» (con la opción `spanish` de `babel`).

Ejercicios

- 9.1 Intente incluir ficheros gráficos de distintos formatos en un documento \LaTeX , utilizando el controlador que se adapte a su visor preferido. Utilice el archivo `ejercicio9.1`, que se encuentra en varios formatos. Si utiliza Mi \TeX y YAP no incluya ninguna opción de `graphicx`. ¿Recibe mensajes de error del compilador? ¿Es capaz de ver los gráficos?
- 9.2 Realice la misma tarea que en el ejercicio anterior pero utilizando ahora el archivo gráfico `ejercicio9.2.pdf` y la opción `pdftex`. Introduzca el parámetro optativo `width` en `\includegraphics`, asignándole un valor de 10 cm (el gráfico de este archivo tiene más de un metro de ancho). Si compila con PD \LaTeX no debe encontrar problema alguno. Varíe la anchura dada; a continuación determine sólo la altura (por ejemplo, también 10 cm); finalmente fije ambas dimensiones, con y sin el parámetro `keepaspectratio`.
- 9.3 Vea el archivo `ejercicio9.3.eps`. Observará que contiene una página con algún texto y un gráfico. Incluya este archivo en un documento \LaTeX , con opción `clip` y sin ella. ¿Qué observa? Si lleva a cabo la misma tarea con el archivo `ejercicio9.3.ps` obtendrá un comportamiento diferente.
Lea, con un editor de texto cualquiera, los dos ficheros: verá que ambos contienen una línea que comienza con ‘`%%BoundingBox:`’; es la línea que indica el tamaño, y los tamaños respectivos son diferentes: en el primer caso (EPS) el tamaño indicado es exclusivamente el del gráfico incrustado (o encapsulado); en el segundo es el tamaño de la página (A4) completa. Visualice los dos archivos con GSVIEW y elija la opción ‘Mostrar Bounding Box’: el contenido propio del archivo aparecerá recuadrado.
- 9.4 Incluya el archivo gráfico `ejercicio9_4.pdf` en un documento \LaTeX utilizando la opción `dvipdfm` de `graphicx`. Ejecute DVIPDFM sobre el archivo DVI generado tras la compilación para obtener un archivo PDF. No debe encontrar ningún problema, ni en la compilación ni en la ejecución de DVIPDFM. Observe que el archivo `ejercicio9_4.bb` se encuentra presente en la misma carpeta que `ejercicio9_4.pdf` (si copió este archivo a su disco, tendrá que hacer lo mismo con el de extensión `bb`).
- 9.5 Realice la misma tarea que en el ejercicio anterior, pero ahora insertando el archivo gráfico `ejercicio9_5.pdf`. Observe los problemas con los que se encuentra. En este caso el controlador busca el archivo de medidas `ejercicio9_5.bb`, pero éste no existe. Ejecute:
`ebb ejercicio9_5.pdf`
para generar el correspondiente `.bb`. Lea el archivo obtenido: debe contener, entre otras, la línea
`%%BoundingBox: 0 0 302 290`

que es la que indica las dimensiones del gráfico. Vuelva a intentar incluirlo en un documento L^AT_EX para verificar que los problemas han desaparecido. Tenga la precaución de copiar el archivo gráfico a su disco antes de intentar ejecutar EBB.

- 9.6 En cualquiera de los documentos generados en los ejercicios anteriores incluya uno de los gráficos como un objeto flotante de tipo figura. Asigne distintos valores al argumento *Posición* y observe cómo cambia el resultado. Realice la misma experiencia incluyendo varias figuras (sean gráficos o, simplemente, texto) y, si es posible, en un documento de cuatro páginas o más. Genere el índice de figuras para este documento.

PARA SABER MÁS

- ▶ Si deseamos incluir un gráfico en un documento L^AT_EX y todavía no poseemos un archivo electrónico con este gráfico debemos decidir qué herramienta utilizar para generarla; en la sección II.4.1 se describe una posible estrategia.
- ▶ Ya hemos visto que en algunas situaciones es necesario convertir un archivo gráfico a otro formato; algunas herramientas para esta tarea, y sus características, se encuentran descritas también en II.4.1.
- ▶ En la sección II.4.4 encontrará una introducción a PSTricks, un magnífico paquete para generar «gráficos lineales» desde dentro de L^AT_EX, con salida PostScript.
- ▶ Algunas herramientas para incluir figuras pequeñas, rodeadas de texto, en II.3.4.4.
- ▶ Sobre opciones globales y otras utilidades (como escalado y rotación de objetos) del paquete graphicx, y otros parámetros de \includegraphics, en II.4.2.
- ▶ Sobre los parámetros que L^AT_EX maneja a la hora de ubicar un objeto flotante, en II.3.4.1.
- ▶ Los antetítulos ‘Figura’ e ‘Índice de figuras’, para los entornos figure y el índice generado por \listoffigures, pueden ser modificados; encontrará cómo hacerlo en las secciones II.1.2 y en II.2.2.
- ▶ Cómo personalizar la leyenda de las figuras, en II.3.4.3.

Órdenes y declaraciones: comandos y entornos



OBJETIVOS

- Comprender el significado de los términos *comando*, *entorno* y *grupo* para LATEX.
- Aprender la sintaxis normal de los comandos y entornos.
- Definir y redefinir comandos y entornos.

NUESTRA comunicación con el compilador se realiza siempre desde el documento fuente incluyendo en él órdenes y declaraciones. Para comprender bien el funcionamiento de T_EX es importante que conozcamos la forma en la que se le dan las órdenes y en qué parte del documento se aplican. En esta lección vamos a describir con más detalle los tres elementos que intervienen al incluir órdenes en un documento fuente y que hemos venido utilizando: *grupos*, *comandos* y *entornos*. Por otra parte, como no siempre tenemos a nuestra disposición un comando o un entorno que automatice una determinada tarea, vamos a aprender a definir nuevos comandos y entornos que faciliten la escritura de nuestros documentos.

§10.1 Los grupos

EN los documentos fuente de LATEX un grupo es cualquier parte del documento bien delimitada. Normalmente, los límites del grupo están visibles: tiene un punto de inicio, determinado por una llave { o por el inicio de un entorno, y un punto final, que vendrá dado por una llave } o por el final del entorno que lo inició. Pero otros grupos están delimitados de forma implícita por la acción de algunos comandos. Veamos un ejemplo de cada tipo:

- En el ejemplo 10.1, limitamos con llaves el grupo donde actúa el comando \itshape para escribir una palabra en **negrita** utilizando tipos de la familia sanserif.
- El cuerpo de un documento fuente (entre \begin{document} y \end{document}) es un ejemplo de un grupo.
- El argumento de un comando como \textit{es un grupo} (véase también la última línea del ejemplo 10.1).

EJEMPLO 10.1

```
¿Cuántas órdenes son necesarias para
escribir la palabra
{\itshape \bfseries \sffamily Sanserif }
en negrita con perfil itálico y utilizando
tipos de esa \textit{\sffamily familia}?
```

Un grupo delimitado por llaves {}

¿Cuántas órdenes son necesarias para escribir la palabra **Sanserif** en negrita con perfil itálico y utilizando tipos de esa familia?

Al escribir un documento sólo definiremos nuestros propios grupos, delimitados por dos llaves, cuando queramos *declarar* algunas propiedades, por ejemplo el tipo de letra, relativas a una parte del texto. Pero casi constantemente estaremos definiendo grupos para L^AT_EX al escribir entornos, argumentos de comandos, etc.

La noción de grupo puede entenderse como la de una unidad de trabajo para T_EX, a lo largo de la cual T_EX realiza numerosas asignaciones (de longitudes, variables, etc.) que son «olvidadas» una vez que el grupo ha finalizado. Así, por ejemplo, los comandos definidos (o redefinidos) dentro de un grupo dejan de existir (o mantienen su definición inalterada) cuando salimos del grupo, y las declaraciones efectuadas dentro de un grupo pierden su validez fuera del mismo (véase el ejemplo 10.1).

Como es natural imaginar, pueden existir grupos dentro de otros grupos. Es importante resaltar que si un grupo comienza antes de que un grupo anterior haya concluido, entonces el nuevo grupo debe finalizar antes que el primero iniciado. En otras palabras, los grupos que tengan partes comunes deben aparecer anidados entre sí (uno dentro del otro), pues de lo contrario se pueden producir resultados inesperados.

Una regla fácil de seguir, que puede evitar errores cuando queramos construir un grupo, es la de escribir simultáneamente el inicio y el final (las llaves {}) del grupo y a continuación retroceder para escribir su contenido. De esta manera evitaremos intersecciones erróneas de grupos o, lo que es peor, dejar algún grupo abierto, provocando un error en la compilación con L^AT_EX.

§10.2 Comandos y entornos

EN las lecciones anteriores, al seleccionar la clase de documento, al usar distintos tipos de letra, o al paginar nuestro documento, hemos ido dando las correspondientes instrucciones al compilador incluyendo órdenes y declaraciones en los documentos fuente de L^AT_EX. Nos hemos estado refiriendo a estas órdenes con el término *comando* y así continuaremos haciéndolo.

Comandos

En la lección 4 hemos tratado de los diez caracteres reservados por T_EX; todos ellos son comandos. También existen algunos comandos cuyos nombres consisten en el carácter \ seguido de otro carácter que no es una letra. Por ejemplo, los utilizados para imprimir los caracteres especiales (v. §4.1), o los comandos \\ y _ (v. §3.1).

El nombre de la mayor parte de los comandos está formado por el carácter \ seguido de una o varias letras. No puede tener espacios en blanco ni caracteres que no sean propiamente letras del alfabeto.

Algunos comandos no llevan argumentos sobre los que actuar, como, por ejemplo, la declaración \itshape que declara el perfil itálico, o el comando \today que incluye la fecha del día de hoy. Normalmente, al escribir estos comandos podemos terminar con un espacio en blanco, aunque también podemos finalizar con un grupo vacío {}, un punto, un paréntesis, uno de los caracteres reservados o cualquier otro comando. Cuando un espacio en blanco sigue inmediatamente a un comando, dicho espacio no es considerado en la compilación. En el ejemplo siguiente podemos observar este efecto.

EJEMPLO 10.2

El logotipo \LaTeX se ha unido con la siguiente palabra aunque hemos dejado un espacio en blanco. Para no tener problemas basta acabar el comando con dos llaves antes del espacio en blanco, como en \LaTeX{}.

El logotipo \TeX se ha unido con la siguiente palabra aunque hemos dejado un espacio en blanco. Para no tener problemas basta acabar el comando con dos llaves antes del espacio en blanco, como en \TeX.

Espacios en blanco después de algunos comandos

Muchos comandos tienen argumentos sobre los que basan su actuación. Estos argumentos pueden ser optativos, por lo que pueden estar o no presentes junto al comando, o ser obligatorios, en cuyo caso deben estar presentes. Por ejemplo, los comandos que definen las unidades de estructura como \section[*TextoToc*]{*Título*} tiene un argumento obligatorio y otro optativo (véase la lección 6).

La sintaxis normal para comandos con argumentos es

\NombreComando [ArgOptativo1] ... [ArgOptativoM] {ArgObligatorio1} ... {ArgObligatorioN}

Cada argumento optativo *ArgOptativo* está situado entre corchetes, mientras que cada argumento obligatorio *ArgObligatorio* se escribe entre llaves. Cada comando tiene una sintaxis específica que determina el orden en el que se deben escribir los distintos argumentos, tanto optativos como obligatorios, aunque lo más usual es que los optativos sean los que aparecen en primer lugar, como en la sintaxis mostrada en el recuadro anterior; desgraciadamente dicha sintaxis no está totalmente unificada en este aspecto. Entre el nombre del comando y sus argumentos puede aparecer cualquier número de espacios en blanco. Cuando un comando requiere un argumento obligatorio y \TeX no encuentra una llave después del nombre que indique el comienzo del argumento, considera que dicho argumento es la primera letra o símbolo que encuentra, por lo que puede producir resultados inesperados en el texto compuesto y, en ocasiones, provocar graves errores durante la compilación.

Dependiendo de la acción ordenada por el comando, sus argumentos pueden presentar algunas restricciones, en el sentido de que la acción requerida puede ser incompatible con la de otros comandos que no deberán aparecer como argumentos del primero. Una acción prohibida en el argumento de un comando es la de inicio de un nuevo párrafo, pues provoca errores como el que sucede en el ejemplo siguiente.

EJEMPLO 10.3

```
\textit{f
Error producido al poner un
segundo párrafo como argumento
de un comando} ! Paragraph ended before \text@command was
complete.
<to be read again>
\par
1.17 segundo párrafo como ...
```

Error al usar más de un párrafo en el argumento de los comandos

Los argumentos optativos pueden presentar dos problemas: el primero surge cuando no se usa este argumento y el siguiente carácter en el texto es un corchete, y el segundo problema aparece cuando en el texto *ArgOptativo* tenemos que escribir un corchete. Los dos problemas se resuelven escribiendo los corchetes dentro de un *grupo*, es decir, en la forma `{ [] } o { [] }`.

A veces los argumentos de algunos comandos no sólo sirven para la composición del fragmento de texto donde aparecen, sino que se usan en otras partes del documento. Por ejemplo, los comandos de estructura (`\chapter`, `\section`, etc.) sirven para definir el contenido de las cabeceras de las páginas y para que *LATEX* construya el índice general. A estos argumentos que viajan a distintas partes del documento se los denomina *argumentos móviles*. Conviene saber que estos argumentos móviles pueden ser fuente de problemas. Aunque no entraremos en los detalles en este punto, al menos sí ilustraremos la situación con un ejemplo. Pensemos qué puede ocurrir al intentar poner un pie de página en el título de un capítulo o sección; este título debe viajar al índice general o a la cabecera de las páginas, por lo que se presentará un problema cuando *LATEX* intente llevar ese pie a dicho índice o a la cabecera. En la lección 13 veremos cómo evitar este caso particular, y a lo largo del libro iremos resolviendo otros problemas similares. En terminología *LATEX* se distinguen algunos comandos como *comandos frágiles*; los problemas antes citados pueden surgir cuando algún comando frágil aparece dentro de un argumento móvil (véase el apartado PARA SABER MÁSde esta lección).

Entornos

Cuando *TeX* procesa un comando con argumentos, antes de ofrecer el resultado debe almacenar en la memoria todos los argumentos. Por eso, cuando los argumentos son grandes, esta tarea requiere mucho tiempo y exige mucha memoria, lo que repercute negativamente en el rendimiento del programa. Por otra parte, tal y como señalábamos en el ejemplo 10.3, los argumentos no pueden contener cualquier elemento válido en *LATEX*. Para estos casos *LATEX* proporciona una herramienta muy eficaz: los *entornos*. Un entorno permite decidir qué acciones tomar cuando se entra dentro de él y qué acciones tomar cuando se sale.

Los entornos en *LATEX* son grupos en los que se utilizan las declaraciones de inicio y final para dar órdenes relativas a su contenido, que puede ser extenso y ocupar muchas líneas del documento fuente. Su sintaxis normal es:

```
\begin{NombreEntorno} [ArgOpt1]...[ArgOptM] {ArgObligatorio1}...{ArgObligatorioN}
Objeto
\end{NombreEntorno}
```

Comienza con el comando `\begin{NombreEntorno}` que establece los procedimientos a seguir al inicio del entorno y termina con el comando `\end{NombreEntorno}` que, a su vez, puede realizar otras actuaciones. Como en el caso de los comandos, los entornos también pueden utilizar argumentos, tanto optativos como obligatorios, y el orden de los argumentos optativos y obligatorios no es siempre el indicado en la sintaxis anterior.

En las lecciones anteriores ya hemos utilizado varios entornos, desde el entorno `document` obligatorio para compilar cualquier documento con el formato L^AT_EX, hasta entornos específicos para alinear texto, como los entornos `center` o `flushright`.

§10.3 Definiendo nuevos comandos y entornos

UNA de las principales virtudes de L^AT_EX es la facilidad con la que se pueden definir nuevos comandos y entornos. La finalidad que se persigue cuando se define un nuevo comando, o un nuevo entorno, es doble: por un lado, se busca simplificar la elaboración de nuestros documentos y, por otro, se pretende facilitar futuras revisiones y modificaciones de los mismos.

Nuevos comandos

Supongamos que vamos a utilizar una familia de tipos particular para resaltar algunas partes del texto pero sin estar demasiado seguros de qué familia utilizar. Suponiendo que queremos usar sanserif con perfil itálico, podemos definir un nuevo comando, digamos `\tipoespecial`, de la siguiente forma:

```
\newcommand*{\tipoespecial}{\itshape\sffamily}
```

De esta manera, cada vez que escribamos `{\tipoespecial texto\/}` en el documento fuente, se seleccionará la familia sanserif itálica para nuestro *texto* (observe que conviene incluir la corrección itálica). Tenemos la facilidad de que si, en cualquier momento, queremos cambiar la familia de tipos sanserif por la typewriter, bastará con buscar la definición anterior y cambiar `\sffamily` por `\ttfamily`, consiguiendo que esa única sustitución produzca el cambio de familia de tipo especial en todo el documento.

Los comandos creados por el usuario también pueden admitir argumentos, tanto obligatorios como optativos, siempre que se definan adecuadamente. Por ejemplo, supongamos que queremos que el comando anterior `\tipoespecial` actúe sólo sobre su argumento. Entonces modificamos la definición en la forma:

```
\newcommand*{\tipoespecial}[1]{\itshape\sffamily #1\/}}
```

Ahora si tecleamos `\tipoespecial{texto sanserif}` tras compilar obtendremos *texto sanserif*. La actuación de este nuevo comando sobre su argumento equivale a escribir las llaves del grupo, la declaración de la familia de tipos y del perfil a usar y el contenido del argumento: *texto sanserif*. Se comporta, pues, «como si» la sentencia `\tipoespecial` se sustituyera por su definición (y #1 por el valor del argumento *texto sanserif*), y, en efecto, esto es lo que ocurre; sin entrar en detalles técnicos, es lo que llamamos «expansión» de un comando.

Si definimos un comando que ya estaba definido, el compilador dará un mensaje de error; para evitarlo habría que definir el comando con `\renewcommand*` en lugar de `\newcommand*`, manteniendo la misma sintaxis.

Los comandos así definidos, mediante `\newcommand*` o `\renewcommand*`, pueden tener hasta nueve argumentos e incluso incluir un valor por defecto para el primero de los argumentos, y sólo para él, que se convertiría así en el argumento optativo.

Por ejemplo, si queremos que nuestro texto especial aparezca resaltado por el tipo de letra y por el tamaño (utilizando el tamaño `\large` por defecto) definiremos:

```
\newcommand*{\tipoespecial}[2][\large]{\#1\itshape\sffamily #2\}}
```

Ahora con la sentencia `\tipoespecial{texto sanserif}`, sin argumento optativo, L^AT_EX producirá *texto sanserif*, mientras que para obtener el mismo *texto sanserif* en letras pequeñas, escribiremos `\tipoespecial[\small]{texto sanserif}`, con la declaración del tamaño como argumento optativo.

L^AT_EX proporciona un tercer comando, `\providetcommand*`, para definir nuevos comandos, cuya diferencia con los anteriores es que mientras que `\newcommand*` define nuevos comandos y `\renewcommand*` redefine comandos ya existentes, `\providetcommand*` sirve para definir nuevos comandos pero sin tener ningún efecto si el comando ya existe. Este nuevo comando puede ser útil para la definición de comandos en ficheros susceptibles de ser incorporados a distintos documentos. La sintaxis general para los tres comandos es la siguiente:

```
\newcommand*{\NombreComando } [NúmArg] [ArgDefecto] {Definición}
\renewcommand*{\NombreComando } [NúmArg] [ArgDefecto] {Definición}
\providetcommand*{\NombreComando } [NúmArg] [ArgDefecto] {Definición}
```

NombreComando es el nombre que queremos asignar a nuestro comando (incluyendo la barra inclinada \), *NúmArg* indica el número de argumentos (si los tiene) y es un número comprendido entre 1 y 9, *ArgDefecto* denota el valor por defecto del argumento optativo que será el primero de ellos y *Definición* contiene la definición de nuestro comando. En esta definición la posición de los argumentos se señala, como hemos hecho en nuestro ejemplo, con el símbolo almoadilla, #, junto con el número correspondiente al argumento.

Cuando se definen o redefinen nuevos comandos debemos tener presente que si se realiza dentro de un grupo esta nueva definición dejará de tener efecto al salir del mismo. Por esa razón, una buena política de actuación es la de situar en el preámbulo del documento principal las definiciones que se quieren utilizar en distintas partes del mismo.

Los comandos pueden utilizarse para tareas «exóticas», en el sentido de que es posible usarlos para fines no habituales. Por ejemplo, supongamos que queremos construir un comando, dependiente de un argumento, que nos permita incorporar notas a nuestro manuscrito en las versiones preliminares; lógicamente, en la versión definitiva dichas notas deben desaparecer. Lo más sencillo es redefinir el comando en la versión final para que no haga nada con el argumento. En las versiones preliminares podemos definir el comando `\nota` del siguiente modo (véase el apartado 13.2 para una explicación del comando `\marginpar`):

```
\newcommand*{\nota} [1] {\marginpar{\#1}}
```

si deseamos que las notas de trabajo aparezcan en el margen. Cuando el documento ya esté finalizado y en versión definitiva, cambiaríamos la definición del comando de la siguiente manera:

```
\newcommand*{\nota}[1]{}
```

De este modo nos evitaríamos tener que borrar las notas una a una y nos aseguraríamos de que todas ellas van a ser *ocultadas* (aunque no serán eliminadas del documento fuente).

Si deseamos definir, o redefinir, comandos con argumentos de modo que alguno de estos argumentos pueda admitir más de un párrafo, entonces debemos utilizar las versiones «sin estrella» de los comandos anteriores:

```
\newcommand{\NombreComando}[NúmArg][ArgDefecto]{Definición}
\renewcommand{\NombreComando}[NúmArg][ArgDefecto]{Definición}
\providecommand{\NombreComando}[NúmArg][ArgDefecto]{Definición}
```

Sin embargo, si alguna vez necesitamos utilizar las versiones «sin estrella», entonces deberíamos pensar que lo que necesitamos realmente es definir un nuevo entorno.

Nuevos entornos

Para definir nuevos entornos o redefinir los existentes, disponemos de los siguientes comandos:

```
\newenvironment*{\NombreEntorno}[NúmArg][ArgDef]{DefEntrada}{DefSalida}
\renewenvironment*{\NombreEntorno}[NúmArg][ArgDef]{DefEntrada}{DefSalida}
```

donde *NombreEntorno* es el nombre del entorno, *NúmArg* es el número de argumentos (si los tiene) y será un número comprendido entre 1 y 9, *ArgDef* indica el valor asignado por defecto al argumento optativo, que será el primero de los argumentos, *DefEntrada* indica las órdenes que se ejecutan antes de entrar en el entorno (es el conjunto de órdenes que ejecuta el comando `\begin{\NombreEntorno}`) y *DefSalida* las que se ejecutan al salir del mismo (con el comando `\end{\NombreEntorno}`).

Una observación muy importante: los argumentos de un entorno sólo se pueden utilizar en la definición de *DefEntrada* con el símbolo # seguido del número del argumento (como en las definiciones de los comandos). Por consiguiente, si los necesitamos en *DefSalida* debemos tener la precaución de guardarlos convenientemente. Ilustraremos con un ejemplo la manera de proceder en estos casos. En el ejemplo 10.4 hemos creado un entorno para escribir citas, de tal forma que el nombre del autor sea un argumento. Es conveniente comentar el código de este ejemplo: para poder escribir el nombre del autor al finalizar la cita, hemos guardado el contenido del argumento en el comando `\Autor` y hemos iniciado un entorno `quote` para nuestra cita; después, en el grupo de declaraciones de salida, cerramos el entorno `quote` y escribimos el contenido de `\Autor` en una línea centrada.

Como en el caso de los comandos, también existen formas «sin estrella» para definir entornos con argumentos, cuando alguno de éstos puede albergar más de un párrafo:

```
\newenvironment{\NombreEntorno}[NúmArg][ArgDef]{DefEntrada}{DefSalida}
\renewenvironment{\NombreEntorno}[NúmArg][ArgDef]{DefEntrada}{DefSalida}
```

EJEMPLO 10.4

```
\newenvironment*{cita}[1]{%
  \newcommand{\Autor}{\#1}%
  \begin{quote}\itshape}%
  {\end{quote}\centerline{\Autor}}%
\begin{cita}{Javier}
Mi carrera ha sido lenta como la del caracol,
pero segura y sólida como su concha.
\end{cita}
```

*Mi carrera ha sido lenta como la del caracol,
pero segura y sólida como su concha.*

Javier

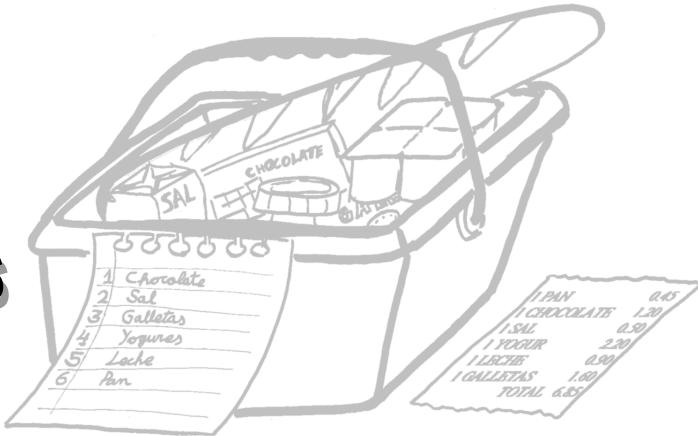
Obsérvese que #1 en la definición del comando auxiliar `\Autor` se refiere al primer argumento del entorno `cita` y que el comando `\Autor` no tiene argumento

Ejercicios

- [10.1] Defina los siguientes comandos sin argumentos: `\rojo`, `\azul`, `\amarillo`, `\verde` y `\grisclaro`, para declarar los correspondientes colores de texto.
- [10.2] Defina un comando `\seccion` con un único argumento obligatorio que actúe como la declaración `\section` para nuevas secciones, escribiendo el título de la sección en color azul y enviando el mismo título al índice general sin color. Intente cambiar el nombre del comando para llamarlo `\sección`: ¿encuentra algún problema?
- [10.3] Defina un entorno `anotacion` que cree un entorno `quote` y escriba el texto usando tipos de la familia `\ttfamily` con tamaño `\small`.
- [10.4] Redefina el entorno del ejercicio anterior para que admita como argumento optativo la fecha, tomando por defecto el valor de `\today`, que aparecerá al final de la anotación justificada al margen derecho de la página.

PARA SABER MÁS

- Contadores y longitudes son parte importante de L^AT_EX. Aunque al lector puedan parecerle comandos, no lo son exactamente; prueba de ello es que para crear o modificar estos registros se utilizan formas distintas a las que hemos visto en esta lección para los comandos y entornos. Encontrará toda la información sobre este tema en la lección 17 y en el apartado II.1.5.
- Hay otras formas de definir comandos, propias de T_EX, no de L^AT_EX, es decir, más básicas, de nivel más bajo; son más difíciles de manejar, pero también ofrecen muchas más posibilidades. Consulte la sección II.8.1.
- En la lección hemos citado los argumentos móviles y los comandos frágiles como fuente de problemas; para saber cómo «proteger» los comandos frágiles, evitando los problemas, vea el apartado II.2.2.2.



OBJETIVOS

- Crear listas numeradas y no numeradas.
- Crear listas descriptivas.

AS listas con apartados o ítems, sean numerados o no, constituyen una forma usual para presentar una información estructurada. A veces tales listas aparecen anidadas entre sí, de tal suerte que alguno de los ítems se desgrana a su vez en una nueva lista de ítems. En esta lección aprenderemos lo sencillo que es construir listas con L^AT_EX sin tener que preocuparnos del mantenimiento de los contadores o de las «sangrías relativas» de los ítems de una lista con relación al formato del texto base en el que aparecen insertadas.

§11.1 Listas numeradas y listas con viñetas



EN las listas numeradas los diferentes ítems están enumerados secuencialmente ya sea con números (arábigos o romanos) o con letras. En las listas no numeradas los diferentes ítems suelen estar identificados mediante una viñeta o marca común. Ambos tipos de listas se construyen en L^AT_EX utilizando sendos entornos cuya estructura sintáctica es análoga.

```
\begin{enumerate}
\item Texto1
\item Texto2
...
\end{enumerate}
```

```
\begin{itemize}
\item Texto1
\item Texto2
...
\end{itemize}
```

El entorno `enumerate` se utiliza para listas numeradas, mientras que el entorno `itemize` se emplea para la generación de listas con viñetas.

Como se aprecia en el ejemplo 11.1, L^AT_EX se ocupa de que el texto de los ítems se imprima dejando una sangría respecto al texto precedente y de numerarlos de forma automática, si se trata de una lista numerada, o de incluir una marca o viñeta, si se trata de una lista sin numerar. De ese

modo, si se intercala un nuevo ítem en una lista numerada todos los ítems vuelven a numerarse en la forma que corresponda.

EJEMPLO 11.1

```
El bosque tropical desaparece ...
\begin{enumerate}
\item Cada 2 segundos se tumba un ...
\item Cada día se destruye un área ...
\end{enumerate}

Para períodos más largos de tiempo, ...
\begin{itemize}
\item Cada año, un área de bosque del
      tamaño de Panamá.
\item Cada 10 años, un ...
\end{itemize}
```

El bosque tropical desaparece a enorme velocidad:

1. Cada 2 segundos se tumba un área de bosque equivalente a un campo de fútbol.
2. Cada día se destruye un área del tamaño de la Ciudad de Guatemala.

Para períodos más largos de tiempo, éstas son las equivalencias:

- Cada año, un área de bosque del tamaño de Panamá.
- Cada 10 años, un área de bosque del tamaño de Venezuela.

 Listas de ítems numeradas y listas de ítems con viñetas

Para facilitar la lectura, el código fuente del ejemplo precedente ha sido escrito en una forma estructurada que incluye sangrías. Esto es solamente una cuestión de confortabilidad para quien escribe, pues L^AT_EX habría producido el mismo resultado aunque todo el código hubiera estado escrito en una sola línea. Conviene, sin embargo, ser ordenado al escribir el texto fuente, y cuidar la organización del mismo; esta adecuada organización es muy beneficiosa, ya que, con toda probabilidad, tendremos que volver a leer el texto fuente en algún otro momento.

Por último, señalemos que el entorno `enumerate` utiliza objetos numerados (los ítems), y, por tanto, el comando `\label` puede ser usado, en la forma indicada en la lección 8, para realizar referencias cruzadas, pudiendo conseguir incluso que estas referencias se transformen en hiperenlaces con el paquete `hyperref` (véase el ejercicio 11.2). El comando `\label` puede ser introducido en cualquier lugar dentro del contenido del ítem a que hace referencia; pero es una buena práctica introducirlo inmediatamente después del comando `\item`.

§11.2 Listas descriptivas

ADEMÁS de los tipos anteriores es necesario, en ocasiones, construir listas en las que cada ítem corresponde a un término o concepto que se formula y se describe, como ocurre en el ejemplo 11.2. Para tales listas L^AT_EX dispone del entorno `description` que sigue. Nótese que las etiquetas se introducen entre corchetes y no entre llaves.

```
\begin{description}
\item[Etiqueta1] Texto1
\item[Etiqueta2] Texto2
\item[Etiqueta3] Texto3
...
\end{description}
```

11.2 Listas descriptivas

EJEMPLO 11.2

```
\begin{description}
    \item[Presidente:] Sus funciones son ...
    \item[Tesorero:] Sus funciones son ...
    \item[Asamblea General:] \hspace{0pt}\par
        Es el órgano supremo ...
\end{description}
```

Listas de ítems con descripciones y un truco

Presidente: Sus funciones son ...

Tesorero: Sus funciones son ...

Asamblea General:

Es el órgano supremo ...

Llamamos la atención sobre el truco empleado para iniciar un nuevo párrafo al comienzo del último ítem. La necesidad de este truco está motivada por la definición del comando `\item` que hubiera ignorado el comando `\par` (o hubiera producido un error con el comando `\newline`) si se hubiera eludido el espacio nulo `\hspace{0pt}`. En lugar de este espacio se puede introducir la sentencia `\mbox{}`, es decir, una caja vacía (véase el apartado 18.1).

Como puede verse, las marcas de los ítems aparecen en negrita en este tipo de listas. Obviamente es posible modificar el formato de letra, ya sea la familia, el perfil, el grosor o el tamaño, indicándolo explícitamente en la etiqueta del ítem, en la forma analizada en la lección 5.

Las listas de ítems que hemos descrito pueden anidarse unas dentro de otras. Así, un ítem de una lista puede contener subítems dentro de él (ítem de segundo, tercero y hasta cuarto nivel), los cuales, a su vez, tendrán una sangría con relación al ítem del que cuelgan y marcas, numéricas o no, que permiten identificar los ítems de los diferentes niveles.

EJEMPLO 11.3

Las cimas destacadas son:

```
\begin{enumerate}
    \item Teide 3\,710 metros
    \item Sierra Nevada
    \begin{enumerate}
        \item Mulhacén 3\,478 metros
        \item Veleta 3\,392 metros
    \end{enumerate}
    \item Cordillera Pirenaica
\end{enumerate}
```

Listas anidadas

Las cimas destacadas son:

1. Teide 3 710 metros
2. Sierra Nevada
 - a) Mulhacén 3 478 metros
 - b) Veleta 3 392 metros
3. Cordillera Pirenaica

En el ejemplo precedente, los ítems de segundo nivel no están identificados con números sino con letras. L^AT_EX tiene sus propias reglas para «numerar» los ítems y si no quedamos satisfechos con su tratamiento podemos modificarlo. El modo adecuado para realizar tales modificaciones es automatizar la numeración y las marcas de los ítems, en la forma descrita en la segunda parte de este libro (consulte las referencias concretas en el apartado PARA SABER MÁS), pero conviene decir también que el argumento optativo del comando `\item` empleado en el entorno `description` está también disponible para los `\item` de los entornos `itemize` y `enumerate`, pudiendo ser utilizado para sobreescribir las marcas automatizadas que generan dichos entornos; evidentemente, se trata en este caso de un proceso manual, con todas sus desventajas frente a uno automático, por lo que debería ser reservado para casos excepcionales.

Ejercicios

- 11.1 Realice distintas experiencias anidando los diferentes tipos de listas y comprobando los resultados obtenidos. Trate de llegar a un nivel de anidado superior a cuatro: ¿qué ocurre?
- 11.2  En el archivo `ejercicio11.2.pdf` encontrará un fragmento de la redacción del articulado de los estatutos de una sociedad. Se ha construido utilizando un entorno `enumerate` y para garantizar que las referencias a los artículos sean correctas se ha hecho uso de los comandos `\label` y `\ref`. Construya el fichero fuente para ese resultado (o uno similar) utilizando referencias cruzadas. Incorpore nuevos artículos y compruebe que las referencias cruzadas son correctas. En alguno de los artículos incorporados inserte una nueva lista numerada y utilice de nuevo los comandos `\label` y `\ref` en la lista de segundo nivel. Compile y compruebe el resultado. Consiga convertir las referencias en hiperenlaces. Si lo desea puede utilizar como punto de partida el archivo `ejercicio11.2.tex` que hemos utilizado para la versión inicial.
- 11.3  En el archivo `ejercicio11.3.pdf` encontrará una lista numerada en la que se ha utilizado el argumento optativo del comando `\item` en el segundo de los apartados. El fichero fuente lo encontrará en `ejercicio11.3.tex`. Sustituya el punto que sigue al número de cada ítem por un paréntesis y compare los resultados obtenidos en ambos casos. Suprima del preámbulo la línea correspondiente al paquete `babel` y vuelva a comparar los resultados. Finalmente sustituya los números de la lista por un menos o guión largo «—».

PARA SABER MÁS

- Si no le gusta el formato con el que `LATEX` numera los distintos niveles de una lista, o las viñetas utilizadas en los ítems, puede modificar dichos formatos; en la sección II.3.2 encontrará más información.
- `LATEX` utiliza varias longitudes para establecer el aspecto de las listas: sangrías, separación vertical entre los ítems, etc. Encontrará una relación de algunas de estas longitudes en la sección II.3.2 y la información necesaria para modificarlas en el apartado 17.2.
- Los diferentes tipos de listas aquí considerados son casos particulares del entorno `list` que se describe en el apartado II.3.2.2. Con ayuda de ese entorno podrá hacer listas completamente personalizadas.

**OBJETIVOS**

- Escribir en columnas.
- Escribir textos paralelos en columnas.

PAQUETES COMENTADOS: `multicol`, `multicolpar`

EL formato de texto en varias columnas es común en los periódicos impresos: la repartición en columnas de los textos periodísticos es una práctica que se impuso en el siglo XVIII. Hay otros tipos de documentos en los que también es habitual encontrar el texto repartido en columnas: diccionarios, enciclopedias, folletos publicitarios, etc. Otras veces en un mismo documento conviven texto en una columna con texto en dos columnas. Por ejemplo, la parte principal de un libro suele estar escrita en una columna mientras que sus índices terminológico y onomástico suelen estarlo en dos. En esta lección explicamos los comandos y paquetes más usuales con los que se puede escribir en varias columnas utilizando **LATEX**.

§12.1 Columnas en LATEX

EN las clases de documento usuales **LATEX** escribe por defecto a una columna. Si queremos que nuestro documento se escriba a dos columnas, como en la figura 9.1, podemos utilizar la opción `twocolumn` al cargar la clase de documento en el preámbulo del texto fuente (véase la lección 6). Si no hemos cargado la opción `twocolumn` y lo que se quiere es que una determinada parte de texto se imprima a dos columnas, se puede hacer uso de los comandos siguientes:

`\twocolumn [Cabecera]`

`\onecolumn`

El primero de ellos provoca un salto de página y cuando el parámetro optativo *Cabecera* está presente imprime el texto *Cabecera* en una línea encima del inicio de las columnas. Todo el texto que se encuentra detrás del comando `\twocolumn` se escribe a dos columnas, hasta que aparezca un comando `\onecolumn` que, de nuevo, produce un salto de página y reinicia la escritura en una columna. Hemos de hacer notar que **LATEX** no se ocupa de equilibrar las columnas antes de realizar el salto de página.

Cuando se ha utilizado el comando `\twocolumn`, o se ha hecho uso de la opción `twocolumn`, el comando `\newpage` inicia una nueva columna. Para iniciar una nueva página es necesario utilizar el comando `\clearpage` (que ubicará los objetos flotantes pendientes).

Las posibilidades de estos comandos para escribir en dos columnas son, como se observa, bastante limitadas, ya que es imposible, por ejemplo, hacer convivir en una misma página texto a una y a dos columnas. Esta posibilidad, entre otras, la ofrecen los paquetes `multicol` y `multicolpar` que describimos a continuación.

§12.2 Columnas con el paquete `multicol`



El paquete `multicol`, desarrollado por Frank Mittelbach [46], permite escribir textos hasta en diez columnas dentro de una misma página, equilibrando la longitud de las columnas para conseguir un efecto estético agradable. Una vez cargado el paquete en el preámbulo de nuestro documento, para escribir en varias columnas se utiliza el entorno `multicols` que sigue:

```
\begin{multicols}{Número} [Cabecera] [Altura]
Texto
\end{multicols}
```

donde los argumentos que aparecen tienen el siguiente significado:

Número Es el número de columnas en que se imprimirá el texto.

Cabecera Es un argumento optativo para un texto común a todas las columnas y que las precede.

Altura Es una longitud que sirve para modificar la altura mínima que debe quedar para completar la página antes de iniciar el entorno `multicols`. Esta altura se guarda en la longitud

```
\premulticols
```

Si la altura que queda para completar la página es menor que el valor de `\premulticols`, se inicia una nueva página antes de empezar a escribir en varias columnas.

Las longitudes

<code>\columnsep</code>	<code>\columnseprule</code>	<code>\multicolsep</code>
-------------------------	-----------------------------	---------------------------

se pueden utilizar para personalizar el entorno `multicols`. La primera es la separación entre columnas. La segunda determina el grosor de la línea de separación entre columnas contiguas; por defecto el valor es 0 pt (la línea de separación es invisible); un valor usual para este tipo de líneas es 0,4 pt. La tercera determina el es-

pacio que separa, por encima y por debajo, un entorno `multicols` del texto que lo circunda; por defecto su valor es 12pt plus 4pt minus 3pt: se trata de una longitud elástica. Estas longitudes pueden modificarse con los comandos explicados en la lección 17.

El texto previo escrito a dos columnas se ha conseguido con un código fuente como

```
\begin{multicols}{2}\columnseprule=0.4pt
se pueden utilizar para personalizar...
\end{multicols}
```

El comando

```
\columnbreak
```

se utiliza dentro de un entorno `multicols` para iniciar una nueva columna. Si se utiliza `\newpage` dentro de un entorno `multicols` se produce un salto de página, a diferencia de lo que ocurre cuando se escribe a dos columnas con `\twocolumn`.

§12.3 Textos paralelos en columnas con el paquete `multicolpar`

EL paquete `multicolpar`, desarrollado por Mauro Orlandini [51], está indicado para imprimir textos en paralelo como los que aparecen en el ejemplo 12.1, recurso utilizado con frecuencia en textos bilingües. El paquete dispone del entorno

```
\begin{multicolpar}{Número}
Párrafos1Columna1 \par
Párrafos1Columna2 \par
Párrafos2Columna1 \par
Párrafos2Columna2 \par
...
\end{multicolpar}
```

donde *Número* indica el número de columnas que se van a utilizar. Mediante el comando `\par` se pasa a la columna siguiente (de forma cíclica) y se inicia un nuevo párrafo. Una línea en blanco produce el mismo resultado. Para iniciar un nuevo párrafo sin que se produzca un salto de columna se utiliza el comando

```
\xpar
```

EJEMPLO 12.1

```
\begin{multicolpar}{2}
TWO roads diverged in a yellow wood,
And sorry I could not travel both.
\par
\noindent Dos caminos se separaban en
un bosque amarillo, Y siento que no
pude caminar por ambos.
\par
Two roads diverged in a wood, and I--
I took the one less traveled by, And...
\par
Dos caminos se separaban en un bosque, y yo--
\end{multicolpar} De \textsc{Robert Frost},...
```

Textos paralelos

TWO roads diverged in a
yellow wood, And sorry I
could not travel both.

Two roads diverged in a
wood, and I—I took the one
less traveled by, And that
has made all the difference.

De ROBERT FROST, “Road not taken”.

Dos caminos se separaban
en un bosque amarillo, Y
siento que no pude caminar
por ambos.

DOS caminos se separa-
ban en un bosque, y yo— Yo
tomé el menos utilizado, Y
esto ha marcado toda la di-
ferencia.

El primer párrafo aparece por defecto sin la sangría habitual, tal y como se observa en el ejemplo anterior: ahí hemos optado por no sangrar tampoco el párrafo correspondiente a la traducción.

El ejemplo siguiente muestra cómo sangrar los párrafos dentro del entorno `multicolpar`.

EJEMPLO 12.2

```
\begin{multicolpar}{2}
\hspace*{\parindent}%
A “cornerstone” of
our thinking is that in the infinitely small
every function becomes linear.
\par
Una «piedra angular» de nuestro
pensamiento es que...
\par
Blessings on him who gladly...
\end{multicolpar}
```

A “cornerstone” of our thinking is that in the infinitely small every function becomes linear.

(J. W. v. GOETHE)

Blessings on him who gladly remembers his forefathers.

(From an unknown mathematical physicist, 1915.)

Una «piedra angular» de nuestro pensamiento es que cada función es casi lineal para magnitudes infinitamente pequeñas.

Bendito aquel que con alegría recuerda a sus antepasados.

Ejercicios

- 12.1** Observe el formato utilizado en el documento `ejercicio12.1.pdf`: fíjese en los detalles. El fichero `codigo12.1.tex` contiene el texto que ha visto (sin formato) y el fichero `guerra12.1.eps` es el gráfico insertado (véase la lección 9). Genere un documento L^AT_EX para obtener lo que ha visto en `ejercicio12.1.pdf`; si no consigue hacerlo, la solución la tiene en `ejercicio12.1_sol.tex`.
- 12.2** Cuando las columnas son estrechas algunos autores optan por prescindir de la justificación en el margen derecho, para evitar la proliferación de guiones de división silábica. ¿Sabe cómo se puede obtener este resultado? Sugerencia: visite la lección 3.

se pueden utilizar para personalizar el entorno `multicols`. La primera es la separación entre columnas; la segunda determina el grosor de la línea de separación entre

columnas contiguas; por defecto el valor es 0 pt (la línea de separación es invisible); un valor usual para este tipo de líneas es 0,4 pt; la tercera determina el espacio que separa, por

encima y por debajo, un entorno `multicols` del texto que lo circunda; por defecto su valor es 12pt plus 4pt minus 3pt.

PARA SABER MÁS

- La inclusión de gráficos en nuestros documentos es algo frecuente. Cuando se quieren incluir gráficos, figuras, tablas, etc., en documentos que están escritos a dos columnas tendremos que decidir si el gráfico se quiere ocupando la anchura total del texto, la anchura de una columna, rodeado por texto, etc. Estas cuestiones y otras sobre cómo incluir gráficos cuando se escribe a varias columnas se tratan en la lección 9.

Lección 13

Notas



OBJETIVOS

- Poner notas a pie de página con L^AT_EX.
- Escribir notas en el margen de las páginas.

AS notas en un escrito son advertencias, explicaciones o comentarios que se sitúan fuera del texto general. Se suelen introducir poniendo una cifra, letra o marca en el texto que sirve de llamada a la nota que se escribirá aparte, normalmente en el pie de la página o en su margen (aunque también pueden ir en otras partes del documento). En esta lección vamos a aprender cómo poner notas a pie de página y al margen con L^AT_EX.

§13.1 Notas a pie de página

INTRODUCIR notas a pie de página con L^AT_EX en un texto normal es una labor muy simple. Basta con utilizar el comando \footnote, descrito a continuación, para que L^AT_EX numere y genere la «marca de la llamada a la nota», ponga una raya de separación entre el texto principal (o normal) y la zona destinada a las notas a pie y envíe el texto de la nota al pie de la página utilizando una letra más pequeña.

```
\footnote[Número]{Texto}
```

Texto es el contenido de la nota a pie de página.

Número es un argumento optativo que se utiliza para indicar explícitamente el número de la nota.

Si no se utiliza este argumento optativo, L^AT_EX usa el valor del contador de las notas a pie, `footnote`, para numerarlas de forma automática (véanse el ejemplo 13.1 y la lección 17 para más detalles sobre los contadores).

Al escribir el argumento *Número* (que debe ser un número entero 0, 1, 2... o un comando que genere uno de estos números), la marca correspondiente a dicho *Número* aparecerá tanto en el texto principal como en el pie de página. El valor almacenado en el contador `footnote` no se modificará tras la inclusión de una nota al pie que utilice este argumento optativo.

El argumento del comando `\footnote` puede estar formado por varios párrafos, siendo una de las excepciones a la regla general, descrita en la lección 10, de que los argumentos de los comandos no pueden contener la orden `\par` ni líneas en blanco, salvo que hayan sido definidos con las versiones no estrelladas. Por otra parte, este comando `\footnote` no se puede anidar, es decir, no puede aparecer en el argumento de otro comando `\footnote`.

EJEMPLO 13.1

```
Éste es un texto que tiene dos
notas\footnote{Aquí está
la primera nota a pie.}
a pie de página
numeradas automáticamente
por \LaTeX\footnote{Y aquí la segunda.}.
```

Éste es un texto que tiene dos notas¹ a pie de página numeradas automáticamente por \LaTeX ².

¹Aquí está la primera nota a pie.

²Y aquí la segunda.

Notas a pie de página

\LaTeX numera las notas de distinta forma dependiendo de la clase de documento elegida. En la clase `article` las notas se numeran consecutivamente a lo largo de todo el documento, mientras que en la clase `book` la numeración de las notas empieza por el valor uno en cada capítulo (el contador `footnote` se vuelve a iniciar en cada capítulo).

Aunque no es lo habitual, en algunos casos se pueden escribir una gran cantidad de notas o notas muy largas (por ejemplo, en ediciones críticas). \LaTeX tiene predefinida una zona de la página para incluir notas a pie. Cuando el espacio que ocupan las notas es mayor que el que se les reserva, el compilador las divide y envía parte de éstas a la página siguiente.

Cuando estamos escribiendo en dos columnas con la opción `twocolumn`, o con el comando `\twocolumn`, las notas a pie de página aparecen al pie de cada una de las respectivas columnas. Pero si las notas al pie de una columna son demasiado extensas, se dividen y continúan bajo la columna siguiente. En cuanto a las notas en columnas creadas con los paquetes `multicol` y `multicolpar` la situación es distinta. Cuando incluimos notas a pie de página en un entorno `multicols` éstas aparecerán al pie de la página, abarcando todas las columnas. Para incluir notas a pie de página en entornos `multicolpar` debemos optar por utilizar los comandos `\footnotetext` y `\footnotemark` descritos en este mismo apartado.

En la lección 10, página 92, prometíamos indicar cómo poner notas a pie de página en los títulos de las unidades de estructura. El problema consistía en que una nota a pie no puede viajar a distintas partes del texto. Para resolverlo, bastará con utilizar el argumento optativo de la declaración de la unidad (véase el apartado 6.2), donde se escriben los títulos que viajan, en los que no se incluyen las notas al pie, dejando la inclusión de estas notas con el comando *frágil* `\footnote` para el argumento obligatorio de las mismas. Por ejemplo, si en un documento escribimos:

```
\section[Notas a pie]%
{Notas a pie\footnote{Nota al título de la sección}}
```

al compilar obtendremos una nota al pie de la página donde comienza la sección, y el argumento opcional (sin nota a pie) habrá viajado al índice general y, posiblemente, a la cabecera de la página.

Si se nos hubiese ocurrido escribir:

```
\section{Notas a pie\footnote{Nota al título de la sección}}
LATEX habría protestado seriamente enviando el siguiente mensaje de error:
! Argument of \@sect has an extra }.
<inserted text>
      \par
1.74 ...te{Nota al título de la sección}}
```

Situaciones especiales en las notas a pie

En ocasiones se requiere hacer varias llamadas desde el texto principal a un mismo texto situado a pie de página (véase el ejemplo 13.2). Para hacer estas llamadas múltiples podemos hacer uso de los dos comandos que siguen:

<code>\footnotemark [Número]</code>	<code>\footnotetext [Número] {Texto}</code>
-------------------------------------	---

donde *Número* y *Texto* tienen el mismo significado que en el caso de `\footnote`.

El primero de estos comandos se ocupa de poner la marca de llamada a pie de página en el texto principal. Utiliza para ello como número de la nota el valor del argumento optativo si está presente o bien, en caso contrario, incrementa en una unidad el valor del contador `footnote` y utiliza ese número.

El segundo imprime el *Texto* en el pie de página y para la marca numérica utiliza el número del argumento optativo, si está presente, o, en caso contrario, el valor del contador `footnote`.

EJEMPLO 13.2

```
\newcommand{\numeropie}{\arabic{footnote}}
Desde esta palabra\footnote{Pie de página a
referenciar en dos puntos}
\let\guardopie\numeropie%
enviamos una nota al pie...

Desde este nuevo
párrafo\footnotemark[\guardopie]
volvemos a apuntar hacia la misma nota
del pie de página.
```

Desde esta palabra¹ enviamos una nota al pie...

Desde este nuevo párrafo¹ volvemos a apuntar hacia la misma nota del pie de página.

¹Pie de página a referenciar en dos puntos

Llamadas múltiples a notas a pie

En el ejemplo 13.2 se muestra cómo utilizar de estos dos comandos para hacer dos llamadas a una misma nota al pie. Obsérvese que definimos un nuevo comando `\numeropie` para representar el número del pie utilizando la expresión `\arabic{footnote}` (véase el apartado 17.1 que describe el uso de los contadores). También hacemos una asignación con el comando `\let` (véase la sección II.8.1.5) para guardar la representación del número de la nota a la que nos referimos dos veces. Esta asignación se realiza justo después de la inclusión de la nota.

Marcas especiales para las notas a pie

Como ya hemos dicho, L^AT_EX utiliza el contador `footnote` para la numeración de las notas a pie y una representación del mismo mediante números arábigos para las «marcas», lo que en general resulta adecuado. Pero si se utilizan estas notas sólo ocasionalmente, puede ser preferible utilizar, en el lugar de los números, otros símbolos. Para conseguir este propósito basta cambiar la representación del contador `footnote` (véase la lección 17). Por ejemplo, si estamos utilizando el paquete `babel` con la opción `spanish` y redefinimos el comando `\thefootnote` que contiene la representación del contador, mediante

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

sólo dispondremos de seis «marcas» para representar a los números del 1 al 6, mediante el mismo número de asteriscos. Si no se usa el paquete `babel` con la opción `spanish`, con la misma redefinición, se pueden representar nueve números mediante símbolos como *, †, ‡, etc.

§13.2 Notas al margen

AS notas al margen o marginales proporcionadas por L^AT_EX aparecen situadas en los márgenes, ¡como debe ser!, y se distinguen de las notas a pie, además de por su ubicación, por que no utilizan marcas para hacer llamadas. Su uso puede ser razonable cuando se hacen anotaciones y comentarios personales acerca del contenido del texto.

La inclusión de notas al margen resulta tan simple como incluir notas a pie. Para realizarla disponemos del comando

```
\marginpar[TextoIzquierda]{TextoDerecha}
```

que crea una nota al margen cuyo contenido, *TextoDerecha* (si se elude el argumento opcional), comienza a imprimirse, en general, a la misma altura en que se imprime la línea del texto principal en que aparece la palabra que precede al comando `\marginpar` (véase el ejemplo 13.3).

EJEMPLO 13.3

```
Poner notas al margen%
\marginpar{; como ésta!}
es sencillo con \LaTeX. En cada página
se reserva un espacio para estas...
```

Poner notas al margen es sencillo con L^AT_EX. En cada página se reserva un espacio para estas notas. El margen utilizado dependerá de las opciones de la clase de documento que utilicemos. ¡como ésta!

Nota al margen

En la lección 10, página 94, hemos utilizado las notas al margen para definir un nuevo comando que permite hacer anotaciones personales en las primeras versiones de un trabajo.

Si no se usa el argumento optativo, *TextoIzquierda*, la nota aparece a la derecha cuando se utiliza la opción `oneside`, en el margen externo cuando se utiliza la opción `twoside` (por ejemplo, en la clase `book`) y en el margen más cercano al lugar en el que aparece el comando si se escribe

a dos columnas con el comando `\twocolumn`¹. Si se utiliza el argumento optativo *TextoIzquierda*, éste será el utilizado cuando la nota aparezca en el margen izquierdo, mientras que *TextoDerecha* será el de la nota cuando ésta aparezca en el margen derecho.

En la cuestión de la elección del lado derecho o izquierdo para la ubicación de las notas al margen tienen una influencia directa las siguientes declaraciones:

`\normalmarginpar`

`\reversemarginpar`

La primera declaración es la usada por defecto (las notas se ubican como se ha comentado en el párrafo anterior). La segunda invierte el comportamiento por defecto.

El uso de `\marginpar` en algunos objetos de L^AT_EX puede provocar efectos no deseados. Así, por ejemplo, en un entorno `multicols` las notas al margen son ignoradas, y en entornos `multicolpar` y en general dentro de objetos como cajas y tablas la inclusión de estas notas produce errores de compilación. Una buena solución para utilizar notas al margen relativas a estos objetos es incluirlas justo antes de entrar en ellos o justo después de terminar de escribirlos.

Ejercicios

- 13.1  En el fichero `ejercicio13.1.tex` se encuentra el texto del artículo en el que el creador de T_EX, Donald E. Knuth, ponía a disposición de la humanidad su programa. Hemos suprimido las notas a pie y las notas marginales que aparecen en `ejercicio13.1.pdf`. Modifique el fichero `ejercicio13.1.tex` para que al compilarlo aparezcan todas las notas.
- 13.2 Trabajando con un documento de la clase `book`, ponga en el margen los ángulos, >>> o <<<, apuntando hacia el texto con independencia de que la página sea par o impar.

PARA SABER MÁS

- ▶ Sobre comandos frágiles y cómo protegerlos para tratar de evitar algunos de los problemas que generan, en II.2.2.2.
- ▶ Sobre cómo incluir notas a pie de página en tablas véase la página 116, y en cajas y entornos `minipage`, véase la página 150.
- ▶ Hay varios parámetros que controlan el aspecto de las notas a pie: el espacio de separación con el cuerpo del texto, la raya de separación, el espacio reservado para ellas, etc. El comando `\footnoterule` y la longitud `\footnotesep` son algunos de los parámetros; éstos y otros son analizados en la sección II.2.5.
- ▶ La apariencia de las notas al margen puede adaptarse a nuestros gustos o necesidades; basta modificar algunas longitudes, como `\marginparwidth`, `\marginparsep` (que se muestran en la figura 7.1) y `\marginparpush`; todas ellas se describen en la sección II.2.5.

¹Si, en lugar del comando, se usa la opción `twocolumn` al declarar la clase de documento, el comportamiento de las notas al margen no es el adecuado porque pueden sobreescibirse encima del texto en una de las columnas.

Lección 14 Tablas



OBJETIVOS

- Construir tablas sencillas.
- Incluir rayas o filetes de separación en las tablas.
- Realizar construcciones más complejas en las tablas: alineación vertical de un elemento y textos que ocupan varias columnas.
- Ubicar las tablas en objetos flotantes: los cuadros.
- Crear el índice de cuadros.

LA construcción de tablas en un documento \LaTeX es, a menudo, una tarea delicada, en parte por la complejidad del propio objeto que se desea escribir, pero también por la flexibilidad que aporta \LaTeX . Es frecuente que los que se inician en el manejo de \LaTeX comparén la generación de tablas en este sistema con la que se realiza en algún editor de textos de tipo WYSIWYG («What You See Is What You Get» o «lo que ves es lo que tienes»). La comparación es, en cualquier tema, inevitable: el ratón frente a la escritura de comandos... No se trata aquí de sumergirnos en esta discusión pero conviene, al menos, afirmar nuestro punto de vista: las tablas no son sencillas de construir con \LaTeX , pero permiten una enorme flexibilidad de diseño y manipulación. Además se han creado numerosos paquetes que expanden las posibilidades de \LaTeX básico en estas construcciones; algunos de éstos serán abordados con detalle en la segunda parte de este libro.

§14.1 Tablas básicas

LA herramienta básica de \LaTeX para la escritura de tablas es el entorno `tabular`, cuya sintaxis presentaremos en primer lugar. A dicha sintaxis fundamental se pueden ir añadiendo elementos adicionales, como rayas o filetes de separación entre filas y columnas, o elementos que se extienden a varias columnas; se obtienen así herramientas suficientes para la creación de tablas de una cierta complejidad.

La sintaxis del entorno `tabular`, para una tabla de, digamos, P filas y N columnas, es la siguiente:

```
\begin{tabular}[Posición]{FormatoColumnas}
Fila1Columna1 & Fila1Columna2 & ... & Fila1ColumnaN \\
Fila2Columna1 & Fila2Columna2 & ... & Fila2ColumnaN \\
... & ... & ... & ... \\
FilaPColumna1 & FilaPColumna2 & ... & FilaPColumnaN
\end{tabular}
```

Antes de describir con precisión los elementos de esta sintaxis conviene saber que:

- toda la tabla (el cuerpo del entorno `tabular`) es tratada como un todo, una «letra gorda» o caja (véase la lección 18), en particular, la tabla no puede ser separada en dos, aunque no quepa en la línea;
- la sintaxis de `tabular` no contiene elementos propios que centren la tabla o inicien nueva línea o párrafo (todos los comandos necesarios para esas acciones deben ser introducidos por el usuario en el lugar adecuado); por ello, la tabla puede ser incluida en una línea de texto usual (lo que, probablemente, supondrá tener una línea de altura muy superior a la normal);
- en principio no hay limitación para el número de columnas y filas, aunque naturalmente pueden aparecer problemas de composición de la página (no de compilación) si la anchura y/o la altura de la tabla sobrepasa las del texto usual en una página;
- el entorno se puede anidar, es decir, un entorno `tabular` se puede incluir en cualquier casilla de otro entorno `tabular`.

Téngase en cuenta que aunque en el recuadro anterior el cuerpo de dicho entorno se haya presentado con los signos `&` alineados verticalmente, esto no es, en absoluto, necesario; ahora bien, dado que L^AT_EX es muy permisivo a la hora de estructurar la información del texto fuente, es, desde luego, conveniente que la escritura sea lo más clara y estructurada posible, pues esto facilitará la búsqueda de errores y su corrección, como ya hemos sugerido en otros lugares (v. pág. 98).

A continuación se detalla el significado de los distintos elementos de la sintaxis del entorno `tabular`.

- & Es el carácter que se utiliza para separar una columna de la columna siguiente en una fila cualquiera: marca el fin de una casilla y el comienzo de la siguiente en la misma fila (recuerde que `&` fue calificado en la lección 4 como carácter reservado). Podemos dejar una casilla vacía ya sea escribiendo dos `&` inmediatamente seguidos (`&&`) o bien con algún espacio entre ellos; el resultado será el mismo.
- \\ Inicia una nueva fila. Todas las filas, salvo la última, deben terminar con este comando. La última fila no necesita terminar con `\` salvo que la siga algún elemento que, normalmente, es una raya de cierre horizontal (véase cómo incluir estas rayas en la página 114). El comando `\` puede ser sustituido, dentro de un entorno `tabular`, por el comando:

```
\tabularnewline
```

Ambos son prácticamente equivalentes. El primero tiene la ventaja de la brevedad, mientras que el segundo aporta ventajas en circunstancias específicas (concretamente en el uso de \raggedright en columnas con formato de tipo p, véase la página 116).

Una fila puede terminar con \\ (o con \tabularnewline) prematuramente, es decir, sin que contenga todos los caracteres & previstos por el número de columnas (incluso sin que contenga ninguno de ellos); el resultado es que todas las casillas posteriores al último & incluido aparecerán vacías.

Posición Es un argumento optativo que especifica la posición de la tabla respecto a la línea en la que se ha incluido. Puede tomar uno de los tres valores siguientes: t, b, c; cada uno de ellos indica qué parte de la tabla estará alineada horizontalmente con el resto del texto en la línea: t significa que esta parte será la parte superior, b la parte inferior y c la central. La alineación se realiza respecto a la denominada «línea base» de la línea de texto, que es la línea imaginaria sobre la que descansan todas las letras. Su valor por defecto, por tanto el valor que toma si no se incluye el argumento, es c.

Los valores indicados por *Fila1Columna1*, etc., indican claramente el contenido de las distintas casillas que forman la tabla. Cada uno de ellos puede ser un objeto cualquiera (texto, gráfico, etc.).

El argumento *FormatoColumnas*

El argumento *FormatoColumnas* es de una importancia capital pues indica, entre otras cosas, el número de columnas que poseerá la tabla. Así el número de valores del tipo *FilaIColumnaJ* debe estar en consonancia con lo indicado por *FormatoColumnas*.

Este argumento debe consistir en una lista formada por dos tipos de elementos: los *especificadores* y los *separadores* de columna. Debe aparecer siempre un especificador para cada una de las columnas de la tabla, los separadores; por el contrario, son siempre optativos.

Cada especificador de columna indica la presencia de una columna y el tipo de alineación vertical (o justificación) del contenido de las casillas de la misma. Los posibles valores para un especificador son l, r, c o p:

- l declara una columna en la que el contenido será justificado a la izquierda;
- r como el anterior, pero el contenido se justificará a la derecha;
- c como los anteriores, con el contenido de cada casilla centrado.

En los tres tipos anteriores la anchura de cada columna se determina automáticamente en función del contenido más ancho de cada una de sus filas. Por ello a veces es necesario hacer uso del especificador que figura a continuación.

p{Ancho} Crea una columna de anchura *Ancho*. El contenido de las casillas que forman una columna iniciada con este especificador se compone como un párrafo ordinario, sin sangría inicial. La primera línea de cada casilla se coloca alineada horizontalmente con el contenido de las otras columnas. Este tipo de columna es el indicado cuando alguna de sus casillas incluye un texto largo que provocaría, con cualquier otro especificador, una columna de un ancho excesivo.

El contenido de los tres primeros tipos de columna no puede incluir un salto de línea (\\\ o \newline) y un comando \par, o una línea en blanco, será ignorado. Por el contrario, las columnas de tipo p{Ancho} se comportan como pequeñas páginas y, por tanto, sus casillas pueden incluir comandos \par o líneas en blanco, así como \newline (no pueden incluir \\ ya que este comando marca el final de la fila en un entorno tabular).

EJEMPLO 14.1

```
Países latinoamericanos con más del 40\% de
población indígena (en millones)\[3mm]
\begin{tabular}{lrrc}
\textbf{País} & \textbf{Total} & \textbf{Indígenas} & \textbf{Porcentaje} \\
\textbf{Indígenas} & \textbf{Porcentaje} \\
Bolivia & 6,9 & 4,9 & 71 \\
Guatemala & 8,0 & 5,3 & 66 \\
Perú & 20,0 & 9,3 & 47 \\
Ecuador & 9,5 & 4,1 & 43
\end{tabular}
```

Países latinoamericanos con más del 40 % de población indígena (en millones)

País	Total	Indígenas	Porcentaje
Bolivia	6,9	4,9	71
Guatemala	8,0	5,3	66
Perú	20,0	9,3	47
Ecuador	9,5	4,1	43

Una tabla sencilla (abierta). Como se observa, se inicia con \begin{tabular}{lrrc}, lo que significa que hay cuatro columnas y que los textos dentro de cada columna están ubicados del siguiente modo: a la izquierda en la primera columna, a la derecha en las dos siguientes y centrado en la cuarta. Observe cómo el texto de las celdas de la primera fila ha sido incluido en un comando \textbf; una declaración \bfseries al principio de la fila sólo habría afectado a la primera celda

En cuanto a los separadores, disponemos de los dos tipos descritos a continuación.

- | Se introduce antes de la primera columna, entre dos columnas, o después de la última, para producir una raya vertical separando las columnas. Puede ser repetido, tantas veces como se desee, en cualquier posición. Utilizar dos separadores juntos (||) producirá un filete doble; el resultado no siempre es satisfactorio (véase a este respecto el apartado PARA SABER MÁS de esta lección).
- ©{Objeto} Suprime el espacio entre columnas e inserta en su lugar el *Objeto* declarado. Este tipo de separador suele ser útil cuando los valores a incluir en dos columnas adyacentes están o deben estar separados de forma natural por un separador particular (coma decimal, guión, etc.) que, además, conviene alinear verticalmente; en este caso lo recomendable es incluir este separador como valor del argumento *Objeto*.

Si el número de columnas de un tipo dado es grande, la escritura de los especificadores y separadores correspondientes resulta pesada. Para estos casos disponemos de la abreviatura * que se puede utilizar, dentro del argumento *FormatoColumnas*, con la siguiente sintaxis:

*{Núm}{FormatoCols}

Con esta sentencia se introducirán *Núm* columnas idénticas, cada una de ellas del tipo especificado por *FormatoCols*. El valor de *FormatoCols* puede ser cualquier combinación de los anteriores especificadores, incluyendo separadores. Por ejemplo, |*{5}{r|} es equivalente a |r|r|r|r|r|, y |*{3}{|p{1cm}|}|| es equivalente a ||p{1cm}|p{1cm}|p{1cm}||

Mientras que el separador | produce una raya vertical, el siguiente comando produce una horizontal que abarca el ancho de la tabla.

\hline

Se coloca siempre al principio de la fila; por tanto, si lo precede otra fila, se coloca inmediatamente después del comando `\\"` que finaliza la fila anterior. Puede ser repetido tantas veces como se desee (de nuevo, el resultado en caso de repetición no siempre es satisfactorio).

EJEMPLO 14.2

```
Países latinoamericanos con más del 40% de
población indígena (en millones)\\[3mm]
\begin{tabular}{|lrrc|} \hline
País& Total& Indígenas& Porcentaje\\ \hline
Bolivia & 6,9 & 4,9 & 71\\
Guatemala & 8,0 & 5,3 & 66\\
Perú & 20,0 & 9,3 & 47\\
Ecuador & 9,5 & 4,1 & 43 \\ \hline
\end{tabular}
```

Países latinoamericanos con más del 40 % de población in-dígena (en millones)

País	Total	Indígenas	Porcentaje
Bolivia	6,9	4,9	71
Guatemala	8,0	5,3	66
Perú	20,0	9,3	47
Ecuador	9,5	4,1	43

La misma tabla que en el ejemplo 14.1, pero ahora cerrada. Observe que la última fila requiere el comando `\\"` al final ya que lo sigue un comando `\hline`

§14.2 Algunos ejemplos, detalles y trucos

COMO se observa en el ejemplo anterior, la coma decimal queda alineada verticalmente gracias a que todas las cifras tienen el mismo número de decimales. Cuando no se da esta coincidencia, que es lo habitual, necesitamos hacer uso del separador `@{Objeto}`. El siguiente ejemplo muestra cómo hacerlo y las ventajas que aporta.

EJEMPLO 14.3

```
\newcommand{\NEG}{\negthinspace}
La siguiente tabla refleja, en \NEG \% con
relación al PNB, ...
\begin{center}
\begin{tabular}{|l|r@{,}l|l|} \hline
Estados Unidos & 4 & 7 & 0,15 \\
España & 1 & 82 & 0,25 \\
Francia & 3 & 4 & 0,63 \\
Suecia & 2 & 8 & 0,98 \\ \hline
\end{tabular}
\end{center}
```

La siguiente tabla refleja, en % con relación al PNB, las inversiones en gastos militares (segunda columna) y de ayuda al desarrollo (tercera columna) en 1993

Estados Unidos	4,7	0,15
España	1,82	0,25
Francia	3,4	0,63
Suecia	2,8	0,98

Alineación vertical del separador decimal mediante el uso de `@`. El comando `\NEG` se ha definido como una abreviatura para `\negthinspace` (a fin de reducir la longitud del párrafo que muestra el código fuente, véase el ejemplo 4.1)

La utilidad de este separador va más allá de la coma decimal, como muestra el ejemplo 14.4

Puesto que el separador `@{Objeto}` suprime el espacio que LATEX introduce entre las columnas, el argumento *Objeto* puede incluir un comando `\hspace{Longitud}`, o cualquier otro comando que introduzca un espacio horizontal (véase §7.3), en caso de necesitar estos espacios. En particular esta estrategia permite modificar el espacio estándar entre dos columnas particulares, sin alterarlo para otras columnas. Un ejemplo de aplicación de este truco es el siguiente: cuando no hay rayas verticales a izquierda o derecha de la tabla LATEX introduce un espacio en su lugar (igual

a la mitad del espacio entre columnas); este espacio puede ser eliminado incluyendo @{} al principio y/o final del argumento *FormatoColumnas*, como en el ejemplo 14.8.

EJEMPLO 14.4

```
\begin{tabular}{|@{\textbf{\quad Capítulo }}r%  
@{:quad Lecciones }r%  
@{\thinspace-\thinspace}l@{\quad}|}  
\hline  
\textbf{1}&8\\  
\textbf{2}&11\\  
\textbf{3}&20\\\hline  
\end{tabular}
```

Otros usos del separador @

Capítulo 1:	Lecciones 1–8
Capítulo 2:	Lecciones 9–11
Capítulo 3:	Lecciones 12–20

Una posibilidad adicional se obtiene con la ayuda del comando

```
\extracolsep{Longitud}
```

donde *Longitud* es cualquier longitud (véase la lección 17). Incluyendo un separador de la forma @{\extracolsep{Longitud}} conseguiremos que todas las columnas posteriores a dicho separador sean precedidas de un espacio adicional de separación igual a *Longitud*. Este espacio adicional sólo puede ser modificado con otro @{\extracolsep{Longitud}}, a partir de una de las columnas posteriores.

En cuanto al tipo de columna p{*Ancho*}, he aquí un ejemplo:

EJEMPLO 14.5

```
\begin{tabular}{|p{4.5cm}|p{2cm}|}  
\hline  
En pomo, cinarrodon ...&  
Árbol\par\vspace*{22pt} Arbusto\\\hline  
En baya o bacciforme ...&  
Árbol\par\vspace{11pt} Arbusto\\\hline  
En drupa o drupáceo ...&  
\raggedright Con 1 hueso\par\vspace{5pt}  
Con 2 o más huesos\tabularnewline  
\hline  
\end{tabular}
```

En pomo, cinarrodon o balausta (participa en su formación el receptor floral y está coronado por el cáliz)	Árbol
	Arbusto
En baya o bacciforme (interior sin endocarpio pétreo, pero a veces con semillas grandes y duras)	Árbol
	Arbusto
En drupa o drupáceo (con endocarpio endurecido)	Con 1 hueso
	Con 2 o más huesos

Columnas de tipo p (párrafo) en tablas. Los espacios verticales (introducidos con \vspace) han sido calculados manualmente, con el fin de separar las dos filas de texto en una misma casilla. Ésta no es, evidentemente, la solución óptima

Si la anchura de una de estas columnas de tipo p no es grande se puede obtener un texto en el que los guiones de división silábica de palabras son excesivamente abundantes. En estos casos el criterio estético suele recomendar la eliminación de la justificación por la derecha. Como hemos aprendido en la lección 3, página 27, esto se consigue mediante el comando \raggedright. Sin embargo este procedimiento no funciona inmediatamente en un entorno tabular. Para conseguir que no se justifique por la derecha debemos introducir \raggedright en la casilla que nos interese no justificar; si dicha casilla es la última de la fila (no necesariamente la situada más a la derecha,

sino la que precede, en el código fuente, al salto de fila), dicha fila debe ser finalizada con el comando `\tabularnewline` en lugar de con el salto ordinario `\backslash`, como en el ejemplo anterior (ésta es la ventaja, anunciada en la página 112, de `\tabularnewline` frente a `\backslash`). Si no se cambia `\backslash` por el comando `\tabularnewline`, la introducción de `\raggedright` provocará un error. Lo que se ha afirmado sobre el comando `\raggedright` vale igual para `\raggedleft` y `\centering`.

En columnas especificadas por `p{Ancho}` se puede producir un pequeño problema en la división silábica de la primera palabra. TeX nunca divide la primera palabra de un párrafo, así que si `Ancho` es una longitud inferior a la de dicha primera palabra, ésta no será dividida. El truco, en tal caso, consiste en iniciar el párrafo con `\hspace{0pt}`, es decir, un espacio de longitud nula (se trata del mismo truco ya utilizado en el ejemplo 11.2 pero, al contrario de lo que allí ocurre, un `\mbox{}` no resuelve el problema); el problema quedará resuelto sin la introducción de espacio visible alguno, si bien el valor estético del resultado será discutible, véase el siguiente ejemplo.

EJEMPLO 14.6

```
\begin{tabular}{|p{3cm}|p{1.6cm}|}\hline
\raggedright Afortunado, dichoso,
            feliz\footnotemark&
\hspace{0pt}bienaventurado\\ \hline
Con buena intención&
            bienintencionadamente\\ \hline
\end{tabular}
\footnotetext{Aceptaciones tomadas del Diccionario de uso del español
de María Moliner}
```

Afortunado, dichoso, feliz ¹	bienaventu- rado
Con buena intención	bienintencionadamente

Notas a pie en tablas y un truco para dividir siládicamente la primera palabra

Para colocar notas a pie de página en el texto de alguna celda de un entorno `tabular` debemos recurrir a los comandos `\footnotetext` y `\footnotemark`, ya presentados en la página 107. En el ejemplo anterior también se ilustra este particular. Naturalmente tendremos que utilizar el argumento optativo *Número* de `\footnotetext` si en el entorno `tabular` aparecen dos o más marcas de pie de página.

§14.3 Textos que se extienden a varias columnas

CON ayuda del comando `\multicolumn` pueden incluirse, en cualquier fila de un entorno `tabular`, textos que se extienden a lo ancho de varias columnas. Uno de los usos más frecuentes de este comando es la creación de una cabecera común a un grupo de columnas, aunque veremos que también aporta mayor flexibilidad al formato de las columnas.

A continuación presentamos la sintaxis del comando y el significado de sus elementos.

```
\multicolumn{Número}{FormatoColumna}{Objeto}
```

Objeto Es el contenido de la casilla que abarcará varias columnas.

Número Es el número de columnas que se reunirán en una única casilla, para albergar al *Objeto*.

Se reunirán las *Número* columnas siguientes, comenzando por aquella en la que se introduce el comando.

FormatoColumna Indica el formato para la casilla múltiple construida por el comando. Su valor es cualquiera de los admitidos en el argumento *FormatoColumnas* del entorno *tabular* aunque, naturalmente, referido a una única columna.

EJEMPLO 14.7

```
\begin{tabular}{|l|r{---}l|r|}\hline
\textbf{Modelo}&
\multicolumn{2}{c|}{\textbf{Color}}&\textbf{Precio}\\
\textbf{Moderno} & Rojo & Verde & 10\,000\\
Clásico & Blanco & Negro & 9\,000 \\ \hline
\end{tabular}
```

Modelo	Color	Precio
Moderno	Rojo—Verde	10 000
Clásico	Blanco—Negro	9 000

Observe en este último ejemplo que el separador | ha sido incluido en el argumento *FormatoColumna* de \multicolumn; concretamente, se ha escrito \multicolumn{2}{c|}{Color}. Con este separador se consigue incluir la raya vertical posterior a la palabra ‘Color’, en la fila correspondiente. De no haber incluido dicho separador no aparecería esta raya de separación entre ‘Color’ y ‘Precio’. La necesidad de incluir el separador en el comando \multicolumn no es un inconveniente, sino una ventaja, ya que esta circunstancia permite utilizar dicho comando para alterar el formato general de una casilla en particular.

Como se observa en el citado ejemplo, el separador incluido es el posterior y no el anterior a la palabra ‘Color’. ¿Por qué es así? La explicación es sencilla, y necesaria; la regla para comprenderlo es: *una columna comienza siempre con un especificador l, r, c... y consta de todo lo que le siga hasta el siguiente especificador, éste excluido*. Hay una excepción natural a esta regla, la que se refiere a la primera columna de un entorno *tabular*: en ésta se incluye todo el material indicado hasta el especificador de la segunda columna, este último excluido.

Cuando se introducen textos que se extienden a varias columnas aparece de forma natural la necesidad de disponer de rayas horizontales que no se extiendan a todo el ancho del entorno *tabular*. Estas rayas pueden introducirse con el comando

```
\cline{x-y}
```

que trazará una raya horizontal desde la columna con número de orden *x* hasta la columna con número de orden *y*, ambas inclusive (véase el ejemplo 14.8).

§14.4 Un nuevo objeto flotante: los cuadros

ENTENDEREMOS en este apartado el término *cuadro* como la forma flotante de una tabla. La terminología no es clara, ni siquiera en los textos especializados en tipografía, y optaremos por la concepción «tabla-cuadro», paralela a la de «gráfico-figura», ya considerada en la lección 9.

Hablaremos, pues, de un objeto flotante destinado a contener tablas, que recibirá el nombre de cuadro, pero que, como el entorno `figure` (figuras), puede contener cualquier cosa. La necesidad de un objeto flotante asociado a las tablas responde, como sucede con los gráficos, a que el tamaño de estos elementos suele ser grande, así como al hecho de que, en general, no está ligado estrictamente a la secuencia del texto y puede ser referenciado en uno o varios lugares del mismo.

EJEMPLO 14.8

```
\begin{tabular}{@{}l@{\extracolsep{12pt}}%
r@{\extracolsep{10pt}}r@{\extracolsep{12pt}}%
r@{\extracolsep{10pt}}r@{}}
\multicolumn{5}{c}{\textbf{\textit{Ayuda oficial al desarrollo}}}
&\hline
&\multicolumn{2}{c}{2000} &\multicolumn{2}{c}{2001}\\
&Mill.\$&\% PNB&Mill.\$&\% PNB\\
Estados Unidos & 9,955&0,10&11,429&0,11\\
Alemania & 5,030&0,27&4,990&0,27\\
España & 1,195&0,22&1,737&0,30\\
Noruega & 1,264&0,80&1,346&0,83\\
\end{tabular}
```

	Ayuda oficial al desarrollo			
	2000	2001	Mill. \$	% PNB
Estados Unidos	9 955	0,10	11 429	0,11
Alemania	5 030	0,27	4 990	0,27
España	1 195	0,22	1 737	0,30
Noruega	1 264	0,80	1 346	0,83

Textos que abarcan varias columnas y bordes parciales en tablas. En este ejemplo se hace uso abundante del separador `@{\extracolsep{Long}}` (véase la página 115); los ejercicios 14.4 y 14.5 le ayudarán a comprender el porqué de este código

El entorno flotante al que nos estamos refiriendo es el entorno `table`, cuya sintaxis es:

```
\begin{table}[Posición]
Objeto
\caption[TextoLeyendaÍndice]{TextoLeyenda}
\end{table}
```

El significado de los distintos elementos de esta sintaxis es exactamente el mismo que en el caso de `figure` y ha sido profusamente descrito en la lección 9 (págs. 84 y ss.). Si se utiliza este comando el entorno `table` correspondiente es numerado; el contador que se encarga de numerar los cuadros es el contador `table`. Conviene asimismo insistir en la conveniencia de incluir una etiqueta en este entorno (con `\label`) para que pueda ser referenciado; en ese caso, de nuevo al igual que en el entorno `figure`, este `\label` debe aparecer dentro del argumento `TextoLeyenda` del comando `\caption` o inmediatamente después de dicho comando.

La diferencia esencial entre los entornos `table` y `figure` es que generan índices distintos. A los entornos `table` se les asocia el índice de cuadros, generado mediante el comando

```
\listoftables
```

De nuevo son necesarias al menos dos compilaciones para tener un índice de cuadros actualizado, siendo la información sobre dicho índice almacenada en el fichero auxiliar de nombre `el` del documento compilado y extensión `lot` (por «`list of tables`»). La unidad generada por este comando lleva el antetítulo ‘Índice de cuadros’ (con la opción `spanish` de `babel`).

Existe también un entorno `table*` que posee una relación con `table` totalmente idéntica a la que posee `figure*` con `figure`.

Ejercicios

- 14.1** Construya la siguiente tabla sin necesidad de escribir «`h`» y «`m`» en cada celda, es decir, de forma que se escriban automáticamente con los valores numéricos:

Salida	6 h 00 m
Control 1	6 h 45 m
Control 2	7 h 15 m

- 14.2**  Construya la siguiente tabla (necesitará hacer uso de `\multicolumn`; puede encontrar la solución en el archivo `ejercicio14.2_sol.tex`):

	Grupo 1	Grupo 2	Grupo 3	Grupo 4
Hombres	324	123	250	210
Mujeres	143	243	286	222

- 14.3**  Construya la siguiente tabla (en el archivo `ejercicio14.3_sol.tex` encontrará la solución):

CONTENIDO DEL CURSO		
Capítulo	Páginas	Resumen
1. Números	1 – 8	Se establece el lenguaje básico y se definen los «conjuntos numéricos» y sus propiedades básicas.
2. Continuidad	9 – 20	Se define el concepto de función continua y se estudian la propiedades de las funciones continuas.

- 14.4** Ponga un borde con rayas verticales y horizontales simples en la parte más externa de la tabla que aparece en el ejemplo 14.8.

- 14.5** Para comprender la necesidad de utilizar `@{\extracolsep{Long}}` en el ejemplo 14.8, elimine de su código todos esos comandos; observará que las rayas horizontales pequeñas se unen para formar una única, efecto que se deseaba evitar al construir el ejemplo.

A continuación elimine, uno por uno, los comandos `@{\extracolsep{??pt}}` en el argumento *FormatoColumnas* del entorno `tabular`: ¿qué resultados obtiene? (preste particular atención a la longitud de las rayas horizontales obtenidas y a la separación horizontal entre columnas). El ejercicio siguiente contiene estructuras similares a las del ejemplo 14.8, mostrando una estrategia distinta para conseguir las.

- 14.6**  Construya la siguiente tabla (la solución en el archivo `ejercicio14.6_sol.tex`):

Grupo	Medida (en % de respuestas correctas)							
	Edad (meses)		Lectura		Sílabas segmentación		Fonemas segmentación	
	M	SD	M	SD	M	SD	M	SD
Experimental	86,9	3,7	82,7	10,2	87,2	10,4	81,6	13,0
Control	89,2	3,1	77,9	16,9	86,5	10,0	82,4	12,4

PARA SABER MÁS

- ▶ El entorno `array` es una estructura de tipo `tabular` que se emplea en matemáticas para, por ejemplo, construir matrices y determinantes, y será descrita en la sección II.5.6.
- ▶ Las columnas de tipo `p{Ancho}` están construidas internamente en términos de un elemento típico de `TEX`: las cajas. Un conocimiento de las cajas permite reproducir estructuras como las de estas columnas en columnas con distinto especificador; véanse para ello la lección 18 y la sección II.3.5.
- ▶ El formato de las tablas está controlado por numerosos parámetros; véase la sección II.3.3.1 para una descripción de los más importantes.
- ▶ Como ya hemos comentado las rayas dobles en un entorno `tabular` no siempre son estéticamente correctas; para introducir mejoras con el paquete `hhline`, véase la sección II.3.3.5.
- ▶ Sobre tablas con color, véase la sección II.3.3.6
- ▶ Sobre tablas con columnas de números decimales con el paquete `dcolumn`, véase el fichero `dcolumn.pdf` del CD-ROM.
- ▶ Cuando las tablas son tan grandes que no caben en una página el problema al que nos enfrentamos es grave; una solución sencilla y perfecta la aporta el paquete `longtable`, véase la sección II.3.3.3.
- ▶ Hemos aprendido cómo introducir textos que se extienden a varias columnas en una fila, pero no qué hacer para introducir textos que comprendan varias filas en una misma columna; ésta es, quizás, la ausencia más llamativa de esta lección; para su solución, véase la sección II.3.3.4.
- ▶ Cómo personalizar la leyenda de los cuadros, en la sección II.3.4.3.
- ▶ El antetítulo ‘Índice de cuadros’ que, por defecto, incluye `\listoftables` con el paquete `babel` y opción `spanish` puede ser cambiado, véase la sección II.1.2.2.

Citas bibliográficas



OBJETIVOS

- Conocer las principales propiedades de las referencias bibliográficas.
- Realizar referencias bibliográficas y listas de bibliografía con L^AT_EX.
- Convertir las citas bibliográficas en hiperenlaces a las correspondientes referencias bibliográficas.

PAQUETES COMENTADOS: color, hyperref

EN la lección 8 hemos analizado las referencias cruzadas que, mediante etiquetas internas, pretenden facilitar al lector la «navegación» por nuestro documento. Las referencias bibliográficas son referencias externas a otros documentos, libros, artículos, etc., que persiguen, entre otros, los siguientes objetivos: atribuir la autoría de una idea o resultado a quien originalmente la publicó, economizar espacio en el documento que se está escribiendo, o proporcionar información para continuar con lecturas complementarias al documento.

Las citas bibliográficas son usuales en los documentos o publicaciones de carácter científico (tanto en las ciencias experimentales como en las humanidades), pues son esenciales para una completa lectura de los mismos ya que, por ejemplo, determinados razonamientos, descripciones de experimentos o demostraciones de resultados no se escriben de nuevo, sino que simplemente se cita el artículo o libro donde se pueden encontrar.

Existen muchas formas de ordenar las listas bibliográficas y, en determinados círculos académicos, se han desarrollado reglas muy precisas en esta área. Algunas de estas reglas son discutidas en el capítulo sobre referencias bibliográficas de [14].

§15.1 Primeros ejemplos de listas bibliográficas

NORMALMENTE la bibliografía aparece al final del documento, a veces ordenada alfabéticamente y numerada, a veces ordenada pero sin etiqueta (como es habitual en el mundo de las humanidades). Cuando la bibliografía es una lista ordenada numéricamente, la cita a un artículo, libro, etc., se hace indicando el número correspondiente. Es fácil comprender que, ordenadas y numeradas las citas, al insertar o suprimir una publicación en la bibliografía, se verá afectada la

Bibliografía

- [1] GOOSSENS, M., MITTELBACH, F. y SAMARIN, A.: *The L^AT_EX Companion*, Addison-Wesley, 1994.
- [2] GOOSSENS, M., RAHTZ, S. y MITTELBACH, F.: *The L^AT_EX Graphics Companion. Illustrating Documents with T_EX and PostScript*, Addison-Wesley, 1997.
- [3] HAGEN, J. y OTTEN, A.: PPCHT_EX: typesetting chemical formulas in T_EX, *TUGboat* **17** (1992), 54–66.

Figura 15.1: Ejemplo de bibliografía

numeración de todas las que la siguen y, por tanto, todas las citas del documento. El procedimiento con el que L^AT_EX produce las citas bibliográficas es similar al utilizado para las referencias cruzadas (véase la lección 8), donde la utilización de *etiquetas* simplifica enormemente la tarea. L^AT_EX puede presentar las listas bibliográficas tal y como se muestra en la figura 15.1.

Las referencias bibliográficas con L^AT_EX pueden producirse de dos formas:

- (1) Leyendo los datos correspondientes a las citas bibliográficas de una lista que se incluye al final del documento. En este caso es suficiente conocer cómo utilizar el entorno `thebibliography`. En esta lección abordaremos esta primera manera.
- (2) Obteniendo los datos correspondientes a las citas bibliográficas de una base de datos de bibliografía, con la ayuda de un programa externo llamado BIBT_EX. Esta posibilidad será abordada en la segunda parte del libro (véase la sección II.2.8.1).

El ejemplo 15.1 muestra cómo se podría haber incluido parte de las referencias bibliográficas que hemos manejado al escribir este libro. Si fijamos nuestra atención en el código fuente del mismo, observamos que la lista bibliográfica ha sido generada con un entorno similar al entorno `enumerate` descrito en la lección 13, en el que el comando `\item` ha sido reemplazado por el comando `\bibitem` que, ahora, necesita un argumento: la *Etiqueta* que identifica de forma única la referencia bibliográfica que le sigue.

EJEMPLO 15.1

```
\begin{thebibliography}{99}
\bibitem{carlisle} David Carlisle, %
  \emph{The If...}, 1995...
\bibitem{companion} Michel Goossens, Frank%
  Mittelbach...
\bibitem{texbook} Donald E. Knuth. \emph{The%
  \TeX book}...
...
\end{thebibliography}
```

El entorno `thebibliography`

Bibliografía

- [1] David Carlisle, *The If...*, 1995...
- [3] Michel Goossens, Frank Mittelbach...
- [4] Donald E. Knuth. *The T_EXbook...*

Tomando como ejemplo la primera referencia bibliográfica que aparece en el ejemplo 15.1

`\bibitem{carlisle} David Carlisle, \emph{The If...}, 1995...`

observamos que la etiqueta `carlisle` identifica el documento que está escrito a su derecha y con ella se almacena el número que le corresponde una vez compilado el documento, que en este caso es el número 1. ¿Cómo podemos referirnos a este documento en el texto? Para esto, debemos utilizar el comando `\cite` tal y como se muestra en el ejemplo que sigue.

EJEMPLO 15.2

`... \textsf{showkeys}` es un paquete diseñado por David Carlisle (véase `\cite{carlisle}`) y como su nombre indica se encarga de mostrarnos las etiquetas...

Referencias a la bibliografía

... `showkeys` es un paquete diseñado por David Carlisle (véase [1]) y como su nombre indica se encarga de mostrarnos las etiquetas...

La sintaxis general del comando `\cite` es

`\cite[Opcional]{Etiqueta}`

donde *Etiqueta* es el nombre de la etiqueta correspondiente al documento que queremos citar y *Opcional* es un texto opcional que podemos incluir para completar la referencia.

EJEMPLO 15.3

La descripción del comando `\texttt{\textbackslash cite}` puede consultarse en `\cite[página 73]{lamport}`...
`\par Los documentos \cite{carlisle,cheng,% companion}` pueden ser encontrados entre el software ...

Referencias simultáneas a varios documentos de la bibliografía, con indicaciones de página

La descripción del comando `\cite` puede consultarse en [6, página 73] ...

Los documentos [1,2,3] pueden ser encontrados entre el software ...

El comando `\cite` puede utilizarse para hacer una cita a varios documentos, como se ilustra en el ejemplo anterior.

Debe tenerse en cuenta que L^AT_EX, al igual que con las demás referencias cruzadas, produce correctamente las referencias bibliográficas después de dos compilaciones. En la primera compilación se guardan en el correspondiente fichero `.aux` los datos necesarios que, en la segunda compilación, se utilizan para realizar las referencias a la bibliografía.

§15.2 Descripción del entorno `thebibliography`

PARA producir la bibliografía de un documento L^AT_EX dispone del entorno `thebibliography`, que genera la lista de referencias bibliográficas tal y como hemos mostrado en el ejemplo 15.1. La sintaxis general para este entorno es

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

A continuación se presenta el significado de los distintos elementos de esta sintaxis.

LongitudMax Es una cadena de caracteres de anchura mayor o igual que la máxima que va a ser impresa en la numeración de la bibliografía. Por ejemplo, si se sabe que no vamos a tener más de 200 referencias bibliográficas, la orden

```
\begin{thebibliography}{999}
```

instruye a L^AT_EX para que reserve un espacio igual a longitud de la cadena 999 para la numeración que aparece a la izquierda de nuestras referencias en la lista de bibliografía.

Etiqueta Se utiliza para identificar la referencia bibliográfica y es el argumento para el uso posterior en las citas mediante el comando `\cite{Etiqueta}`.

Texto Es el contenido de la cita bibliográfica que normalmente contiene: autor o autores, título del documento, editor o revista, año de publicación, y número de páginas si es relevante (véase la figura 15.1 en la página 122).

Leyenda Se utiliza para modificar la identificación que, en la lista de referencias, se imprimirá a la izquierda de las mismas. Como ya se ha comentado, por defecto L^AT_EX numera las referencias. Si en una entrada

```
\bibitem[Leyenda]{Etiqueta} Texto
```

llenamos el campo opcional *Leyenda*, esta *Leyenda* será la que se imprima en lugar del número de orden correspondiente. Cuando vayamos a utilizar esta opción debemos modificar la longitud *LongitudMax* hasta alcanzar la longitud máxima de todos los argumentos *Leyenda*.

Obsérvese la utilización de *LongitudMax* y del campo opcional *Leyenda* en el ejemplo 15.4, comparándolo con el ejemplo 15.1, donde se encuentran las mismas entradas bibliográficas pero identificadas de otra forma.

EJEMPLO 15.4

```
\begin{thebibliography}{9999999}
\bibitem[Cars95]{carlisle} David Carlisle,%
  \emph{The If...}, 1995...
\bibitem[GMS94]{companion} Michel Goossens,%
  Frank Mittelbach...
\bibitem[Knut86]{texbook} Donald E. Knuth.%
  \emph{The \TeX book}...
\bibitem[Knu86-2]{metabook} Donald E. Knuth.%
  \emph{The Metafont book}... ...
\end{thebibliography}
```

Bibliografía

- | | |
|-----------|--|
| [Cars95] | David Carlisle, <i>The If...</i> , 1995... |
| [GMS94] | Michel Goossens, Frank Mittelbach... |
| [Knut86] | Donald E. Knuth. <i>The TeXbook...</i> |
| [Knu86-2] | Donald E. Knuth. <i>The Metafont book...</i> |
| [Lamp85] | Leslie Lamport. <i>L^AT_EX – A document...</i> |

Referencias por etiquetas con `thebibliography`

§15.3 Citas y enlaces de hipertexto

COMO ya indicábamos en la lección 8, el paquete `hyperref` transforma las referencias cruzadas en enlaces de hipertexto. Y no sólo eso, sino que también las citas bibliográficas son convertidas automáticamente en hiperenlaces, que conectan la cita bibliográfica con el ítem citado de la lista de bibliografía. Por lo que respecta a la bibliografía, las siguientes opciones de `hyperref`, que deben activarse incluyéndolas dentro del comando `\hypersetup`, le añaden una funcionalidad muy interesante:

`backref` Se añade, al final de cada entrada de la bibliografía, una lista con los números de las secciones en las que se cita. Esta opción sólo funciona correctamente si después de cada entrada `\bibitem` existe una línea en blanco.

`pagebackref` Se añade, al final de cada entrada de la bibliografía, una lista con los números de las páginas en las que se cita.

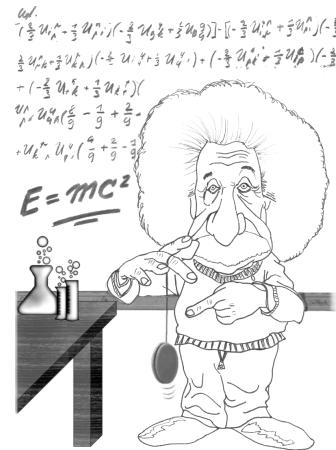
Ejercicios

- 15.1 Examine el archivo `ejercicio15.1.tex` para ver un ejemplo simple de utilización del comando `\cite`. Compile una vez el archivo con PDFLATEX y verifique la salida. Vuelva a compilar y compruebe que la salida es la que aparece en el archivo `ejercicio15.1.pdf`.
- 15.2 En el documento anterior, incorpore el paquete `hyperref` y vuelva a compilar. ¿Observa cómo las citas se han convertido en enlaces de hipertexto?

PARA SABER MÁS

- ▶ Existen muchas formas diferentes tanto de producir una lista de bibliografía como de realizar las citas a los ítems de la misma. Puede consultar algunos de los estilos más usuales en la sección II.2.8.1.
- ▶ Cuando escribimos muchos documentos que comparten bastantes ítems de bibliografía, lo más adecuado es almacenar las referencias bibliográficas en una base de datos y utilizar una herramienta para extraer, en cada caso, los ítems que se citan. El diseño de bases de datos de bibliografía y su utilización se describe en la sección II.2.8.2.
- ▶ El antetítulo de la lista de bibliografía que proporciona el paquete `babel`, con opción `spanish`, es ‘Bibliografía’ para la clase `book` y ‘Referencias’ para `article`; estos valores pueden ser personalizados. Para una lista detallada de todos los títulos que se pueden personalizar, véase la sección II.1.2.2.
- ▶ El paquete `hyperref` permite personalizar el color de las referencias bibliográficas, tanto si se ha activado la opción `colorlinks` como si no. Consulte la página 406 para más detalles.

Fórmulas: una introducción



OBJETIVOS

- Declarar el modo matemático.
- Escribir algunas fórmulas matemáticas sencillas: superíndices y subíndices, fracciones, raíces, etc.
- Incluir textos y espacios en fórmulas.
- Conocer los comandos para escribir las letras griegas.
- Escribir algunas fórmulas químicas sencillas.

PAQUETES COMENTADOS: `amsmath`, `chemsym`

La primera acepción que el *Diccionario de la Lengua de la Real Academia Española* asigna a la palabra *fórmula* es: «medio práctico propuesto para resolver un asunto controvertido o ejecutar algo difícil». Otras acepciones son: «ecuación o regla que relaciona objetos matemáticos o cantidades; combinación de símbolos químicos que expresa la composición de una molécula». El título de esta lección alude, por supuesto, a fórmulas matemáticas, físicas, químicas, etc.

Escribir fórmulas mezcladas con texto no es fácil: el resultado producido por algunos procesadores de texto cuando tienen que acometer esta tarea es penoso. \TeX fue diseñado específicamente para escribir matemáticas de la forma menos desgradable posible. Es fácil escribir con \TeX expresiones científicas o técnicas en general. A \TeX se le dictan las fórmulas. En este sentido, por ejemplo, para escribir una raíz cuadrada se le dice que se trata de una raíz cuadrada y luego se le da el radicando. Una vez que conoce el radicando, \TeX pone el símbolo de la raíz cuadrada de acuerdo con el tamaño del mismo. Lógico, ¿no?, y fácil para el usuario. No hay que preocuparse de los tamaños de las cosas, \TeX hace muy bien ese trabajo.

En esta lección nos ocupamos de describir cómo se dictan las fórmulas utilizando comandos de \LaTeX . Aunque muchos comandos para escribir fórmulas no requieren el uso de ningún paquete adicional, nosotros recomendamos que se cargue el paquete `amsmath`: el paquete incorpora algunos comandos y funcionalidades que hacen más fácil la escritura de fórmulas en nuestro documento. En general no distinguiremos entre los comandos intrínsecos de \LaTeX y los comandos proporcionados por el paquete `amsmath`.

En el capítulo 5 de la segunda parte del libro completaremos las herramientas presentadas en esta lección, mostrando cómo incluir símbolos, operadores matemáticos o delimitadores y manejando otras estructuras y fórmulas más complejas. Si en términos generales L^AT_EX ya le ha podido sorprender, al escribir matemáticas con él sentirá toda la potencia y flexibilidad que hay detrás de sus comandos.

§16.1 El modo matemático

CUANDO T_EX está escribiendo una fórmula trabaja en modo matemático. En este modo T_EX se vuelve un perfeccionista y trata los espacios de forma exquisita. T_EX maneja el modo matemático de dos maneras distintas:

- Modo matemático ordinario. En este modo T_EX compone la fórmula matemática sabiendo que va a estar dentro de un párrafo, por lo que intenta que la altura total de la línea en la que esté no sea demasiado distinta a la del resto de las líneas del párrafo. Así, por ejemplo, utiliza operadores de un tamaño reducido y los subíndices y superíndices están colocados de forma especial.
- Modo matemático resaltado. Es la forma en la que T_EX trabaja cuando queremos que la fórmula aparezca centrada y resaltada. Entonces se utilizan operadores de un tamaño mayor y los espacios se manejan de forma totalmente distinta.

Así, para escribir una fórmula lo primero que hacemos es indicar a L^AT_EX que vamos a entrar en modo matemático y que lo que escribimos a continuación, y hasta que le indiquemos lo contrario, es una fórmula. Para escribir una fórmula dentro de un párrafo junto con texto normal (modo matemático ordinario) podemos utilizar una de las tres opciones que siguen:

$\$ Fórmula \$$	\begin{math}
$\backslash(Fórmula \backslash)$	$Fórmula$

Una recomendación: al declarar una fórmula teclee las expresiones que indican el comienzo y el final de la misma (por ejemplo, los dos signos \$ seguidos), vuelva el cursor hacia atrás y entonces teclee la fórmula; de esta forma nunca olvidará cerrar el modo matemático.

En las formas anteriores de introducción del modo matemático ordinario, la versión entorno `math` y `\(...\)` son totalmente equivalentes. La versión `$...$` es más «primitiva»; es decir, las otras dos están construidas en base a ella. Esto significa que utilizando el entorno `math` o `\(...\)`, posteriormente podemos redefinir ambos globalmente, para que su comportamiento cambie en alguna forma que deseemos, algo que no es posible, y sobre todo no recomendable, con `$...$`.

EJEMPLO 16.1

Si `f` es una función real y `a` es un número, la función `\emph{trasladada}` se define como `f_{a}(x)= f(a+x)`.

Si f es una función real de variable real y a es un número, la función *trasladada* de f se define como $f_a(x) = f(a+x)$.

Modo matemático ordinario

\[Fórmula \]

```
\begin{displaymath}
Fórmula
\end{displaymath}
```

El entorno `displaymath` y la forma `\[...\]` son totalmente equivalentes y se utilizan para declarar el modo matemático resaltado. Las fórmulas resaltadas se imprimen en una línea aparte centrada, salvo cuando se tiene activa la opción `fleqn` en la clase de documento (véase la sección II.5.2), en cuyo caso aparecerán justificadas a una distancia predeterminada del margen izquierdo del texto. Algunos usuarios echarán de menos, entre las opciones que ofrecemos, la forma `$$...$$`, que no recomendamos (véase la sección II.5.1).

EJEMPLO 16.2

Si f es una función real de variable real y a es un número, la función \emph{trasladada} de f se define como $[f_a](x) = f(a+x)$.

Si f es una función real de variable real y a es un número, la función *trasladada* de f se define como

$$f_a(x) = f(a+x).$$

Modo matemático resaltado

Obsérvese que el punto que finaliza el párrafo en el ejemplo anterior es parte de la fórmula resaltada en contraposición con lo que hicimos en el ejemplo 16.1; si en este segundo ejemplo hubiéramos dejado el punto después de \], que cierra el modo matemático resaltado, el resultado hubiera sido desastroso, como muestra el siguiente ejemplo.

EJEMPLO 16.3

Si f es una función real de variable real y a es un número, la función \emph{trasladada} de f se define como $[f_a](x) = f(a-x)$

Si f es una función real de variable real y a es un número, la función *trasladada* de f se define como

$$f_a(x) = f(a+x)$$

Cuando se manejan documentos científicos, las fórmulas resaltadas suelen aparecer numeradas para poder ser referenciadas dentro del texto. Si queremos escribir una fórmula resaltada y que L^AT_EX se encargue de numerarla, disponemos del entorno:

```
\begin{equation}  
Fórmula  
\end{equation}
```

El entorno `equation` numera automáticamente las ecuaciones utilizando para ello el contador `equation` (véase la lección 17 si se quiere aprender a cambiar el valor y la representación de los contadores). Si etiquetamos un entorno `equation` con `\label{Etiqueta}` podemos después referirnos a la misma con los comandos `\ref{Etiqueta}` y `\pageref{Etiqueta}`, como se ha explicado en la lección 8. Además disponemos del comando

\egref{Etiqueta}

que proporciona la misma funcionalidad que `\ref{Etiqueta}`, pero además encierra el número de la ecuación referida entre paréntesis.

EJEMPLO 16.4

```
En la mecánica newtoniana dos cuerpos de masas
\$m\$ y \$M\$ se atraen según una fuerza dada
por la fórmula
\begin{equation}\label{Gravedad}
F=G \frac{m M}{d^2}.
\end{equation}
La ecuación \eqref{Gravedad} es clave para...
```

El entorno equation

Existe una versión `equation*` del entorno `equation` que no numera las ecuaciones y es equivalente a `\[...\]`.

Es importante tener presente que las declaraciones de los tamaños no funcionan dentro del modo matemático, mientras que sí lo hacen las de color. Sin embargo, las declaraciones de tamaño efectuadas en un grupo sí afectan a las fórmulas que son parte del mismo. El ejemplo que sigue ilustra el comportamiento de las declaraciones de tamaños y colores en fórmulas.

EJEMPLO 16.5

```
Albert Einstein nos obsequió con la
siguiente ley:
\[\LARGE E=\color{gray}m c^2.\]
```

La fórmula de la energía potencial
gravitatoria es:
\[\LARGE E_p=mgh.\]

Tamaños y colores de los tipos en las fórmulas

En la mecánica newtoniana dos cuerpos de masas m y M se atraen según una fuerza dada por la fórmula

$$F = G \frac{m M}{d^2}. \quad (16.1)$$

La ecuación (16.1) es clave para...

Albert Einstein nos obsequió con la siguiente ley:

$$E = mc^2.$$

La fórmula de la energía potencial gravitatoria es:

$$E_p = mgh.$$

En modo matemático `TeX` opera de acuerdo con las siguientes normas:

- Utiliza un tipo de letra italicizado en las fórmulas. En algunas circunstancias, según los paquetes que gestionan las familias de tipos que se hayan cargado, este tipo puede ser distinto de la itálica normal, es decir, exclusivo de las fórmulas (esto es así, en particular, con la familia de fuentes Computer Modern, que son las que, por defecto, utiliza `TeX`).
- No respeta los espacios entre palabras y sólo deja espacios cuando él cree que tiene que hacerlo.
- Las fórmulas en modo matemático ordinario son divididas entre líneas de forma natural, cuando es necesario. Además se puede forzar un salto de línea en una fórmula que está dentro de un párrafo (modo matemático ordinario) al recibir el comando `\backslash\backslash`. No permite forzar este salto en el modo matemático resaltado.
- No permite escribir vocales acentuadas dentro del modo matemático; tampoco el carácter ñ.
- Se ocupa de elegir los tamaños adecuados de la raya de fracción, de los subíndices, etc. Estos tamaños son distintos según esté en modo matemático ordinario o resaltado.

§16.2 Fórmulas sencillas

En este apartado explicamos cómo se pueden incorporar superíndices, subíndices, raíces y fracciones a nuestras fórmulas mediante los comandos adecuados de L^AT_EX. En este primer contacto con las fórmulas nos ocupamos de situaciones sencillas y dejamos para una descripción posterior aspectos complicados de las mismas.

Es muy razonable pensar que una expresión tan sencilla como $1,5 - 0,5 + 1 = 2$ debemos escribirla en nuestro documento como $\$1.5-0.5+1=2$$, es decir, como una fórmula y no en modo de texto ordinario como $1.5-0.5+1=2$, que proporcionaría la salida $1.5-0.5+1=2$. Obsérvense las similitudes y diferencias: para las cifras se utilizan los mismos perfiles (aunque esto puede depender de las fuentes utilizadas) pero los signos de sustracción $-$, adición $+$ e igualdad $=$ son distintos; el tratamiento de los espacios es distinto y si escribimos un número decimal, como por ejemplo $\$1.5$$, dentro del modo matemático, el separador decimal se cambia automáticamente a una coma, se escribe $1,5$, siempre que tengamos cargado el paquete `babel` con la opción `spanish`. Para cambiar este comportamiento del separador decimal consulte el apartado PARA SABER MÁS de esta lección.

Si usted es amante de respetar las tradiciones y le gustan los números clásicos como $3,1415$, puede escribirlos utilizando en modo matemático el comando

`\oldstylenums{Número}`

Superíndices y subíndices

Los comandos que se utilizan para escribir superíndices (exponentes) y subíndices son:

`^ {Superíndice}` `_ {Subíndice}`

Para indicar que queremos que una expresión esté como exponente se utiliza acento circunflejo `^` y, a continuación, entre llaves, la expresión *Superíndice* que queremos que figure como exponente. Para un subíndice se emplea el símbolo `_` (barra de subrayar) en lugar del acento circunflejo. Superíndices y subíndices se pueden reiterar y combinar entre ellos.

EJEMPLO 16.6

```
Si $a_{\{0\}}+a_{\{1\}}z+\cdots+a_{\{n\}}z^{\{n\}}+\cdots$ converge para todo número complejo $z$, entonces también lo hace la serie  
$\begin{aligned} &[ a_{\{1\}}z+a_{\{3\}}z^{\{3\}}+ \\ &\cdots+a_{\{2^{\{n\}}+1\}}z^{\{2^{\{n\}}+1\}}+\cdots] \end{aligned}$
```

Superíndices y subíndices

Si $a_0 + a_1 z + \cdots + a_n z^n + \cdots$ converge para todo número complejo z , entonces también lo hace la serie

$$a_1 z + a_3 z^3 + \cdots + a_{2^n+1} z^{2^n+1} + \cdots$$

En el ejemplo anterior hemos utilizado el comando

`\cdots`

que se emplea, como se observa en el ejemplo, para conseguir puntos suspensivos a una cierta altura, adecuada a algunos operadores habituales, como $+$ y $-$ (consideraciones sobre los distintos

tipos de puntos suspensivos y su utilización pueden consultarse en §4.4, para el texto ordinario, y II.5.5.2, para el modo matemático).

Raíces

Aunque una raíz cuadrada (o n -ésima) puede escribirse como un exponente fraccionario, suele expresarse utilizando el símbolo de la raíz cuadrada, $\sqrt{}$. Mediante el comando

`\sqrt[n]{Radicando}`

se obtiene la raíz de orden n de la expresión *Radicando*.

EJEMPLO 16.7

Quienquiera que tenga una calculadora...
valores grandes de n los valores de
 $\sqrt[n]{n}$ están muy cerca de 1.

Quienquiera que tenga una calculadora científica ha podido constatar que para valores grandes de n los valores de $\sqrt[n]{n}$ están muy cerca de 1.

Raíces

Las raíces también se pueden anidar unas dentro de otras, como se muestra en el siguiente ejemplo:

EJEMPLO 16.8

```
{\Large
\[\sqrt{2+\sqrt{2+\sqrt{2}}}\]}
```

$$\sqrt{2 + \sqrt{2 + \sqrt{2}}}$$

Fracciones y números combinatorios

Una fracción, tal y como se nos enseñaba en la escuela, es un objeto con tres elementos: el numerador, el denominador y la barra horizontal que los separa. El comando para escribir fracciones es:

`\frac{Numerador}{Denominador}`

El argumento *Numerador* es la expresión de la fórmula que aparecerá sobre la raya de la fracción y el argumento *Denominador* aparecerá bajo la misma.

EJEMPLO 16.9

La derivada de la expresión
 $\frac{z}{n(n+z)}$
es

$$\frac{1}{n^2(n+z)^2}$$

La derivada de la expresión $\frac{z}{n(n+z)}$ es

$$\frac{n^2}{n^2(n+z)^2}$$

Fracciones

Este ejemplo invita a hacer varios comentarios. Primero, referente a la parte del código. No es relevante escribir la fórmula centrada, como se ha hecho (utilizando marcas de tabulación), ya que si se hubiera escrito todo en una línea la compilación con LATEX hubiera producido el mismo resultado; sin embargo, una escritura como la que se ha utilizado (similar a la que utilizan los programadores cuando escriben sus programas) ayuda a una lectura cómoda del código fuente. Segundo, obsérvese que el tamaño de las fracciones viene determinado automáticamente por el tipo de fórmula en la que aparece (fórmula dentro de un párrafo de texto o fórmula centrada). Obsérvese también que la primera fracción que aparece no es muy legible. En un texto de matemáticas se hubiera escrito:

EJEMPLO 16.10

La derivada de la expresión $z/n(n+z)$ es

$$\begin{aligned} \backslash [\\ \quad \backslash \text{frac}\{n^{\{2\}}\}{n^{\{2\}}(n+z)^{\{2\}}} \\ \backslash] \end{aligned}$$

La derivada de la expresión $z/n(n+z)$ es

$$\frac{n^2}{n^2(n+z)^2}$$

Alternativamente, el tamaño de la fórmula $\frac{z}{n(n+z)}$ se puede incrementar hasta $\frac{z}{n(n+z)}$, mediante la orden $\text{\displaystyle\frac{z}{n(n+z)}}$.

En general, los comandos

\displaystyle	\textstyle
------------------------	---------------------

hacen, respectivamente, que el tamaño de una fórmula que forma parte de un párrafo sea el correspondiente al de esa fórmula cuando está centrada, y viceversa, que el tamaño de una fórmula centrada sea el correspondiente al de esa fórmula cuando es parte de un párrafo.

EJEMPLO 16.11

$$\begin{aligned} & \backslash [\backslash \text{frac}\{1\}{n^{\{2\}}}\backslash \log_2 n^{\{2\}} \\ & \quad \backslash \text{textstyle}\backslash \text{frac}\{1\}{n^{\{2\}}}\backslash \log_2 n^{\{2\}}\backslash] \\ & \$\sqrt{\backslash \text{frac}\{1\}{n^{\{2\}}}\backslash \log_2 n^{\{2\}}}\backslash \sqrt{\backslash \text{displaystyle}\backslash \text{frac}\{1\}{n^{\{2\}}}\backslash \log_2 n^{\{2\}}\$} \end{aligned}$$

$$\begin{aligned} & \frac{1}{n^2} \log_2 n^2 \frac{1}{n^2} \log_2 n^2 \\ & \sqrt{\frac{1}{n^2} \log_2 n^2} \sqrt{\frac{1}{n^2} \log_2 n^2} \end{aligned}$$

Modificando el tamaño de las fórmulas

Cuando tenemos cargado el paquete `amsmath` los comandos

$\text{dfrac}\{Numerador\}{Denominador}$	$\text{tfrac}\{Numerador\}{Denominador}$
--	--

son, respectivamente, formas abreviadas de las expresiones

$\text{\displaystyle\frac{Numerador}{Denominador}}$ y
 $\text{\textstyle\frac{Numerador}{Denominador}}$.

Los números combinatorios que, como las fracciones, también tienen numerador y denominador, se pueden escribir mediante los comandos que se relacionan a continuación.

`\binom{Numerador}{Denominador}`
`\dbinom{Numerador}{Denominador}`

`\tbinom{Numerador}{Denominador}`

Los dos últimos funcionan, como los comandos `\dfrac` y `\tfrac`, modificando el tamaño de la correspondiente expresión binomial.

EJEMPLO 16.12

```
\noindent El número de subconjuntos de  
un conjunto de $n$ elementos es:  
$\left[ 2^n = (1+1)^n = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{n} \right]
```

El número de subconjuntos de un conjunto de n elementos es:

$$2^n = (1+1)^n = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{n}.$$

Números combinatorios

§16.3 Inserción de textos y espacios en fórmulas

FÓRMULAS y texto conviven en general, o pueden convivir, en un documento. Una vez que hemos declarado el modo matemático mediante, por ejemplo, un `$`, todo lo que se encuentre hasta el siguiente `$` se procesa, como ya se explicó al final del apartado 16.1, ignorando espacios y en una fuente de tipo itálico. Si nos fijamos en cualquiera de los ejemplos de esta lección, en los que conviven texto y fórmulas, hemos abierto el modo matemático al comienzo de una fórmula y lo hemos cerrado cuando ésta termina exactamente. Si en el ejemplo 16.1 no hubiéramos abierto y cerrado el modo matemático con «sensatez», el resultado podría haber sido algo como lo que sigue:

EJEMPLO 16.13

Si `f` es una función real de variable real y `a` es un número, la función `\emph{trasladada}` de `f` se define como `f_{a}(x) := f(a+x)`.

Si f es una función real de variable real y a es un número, la función *trasladada* de f se define como $f_a(x) := f(a+x)$.

En algunos casos puede que sea necesario no cerrar el modo matemático e incluir en él, como parte de una fórmula, un fragmento de texto. El ejemplo que sigue ilustra esta situación.

EJEMPLO 16.14

```
Si $f$ es una función real de variable  
real y $a$ es un número, la función  
\emph{trasladada} de $f$ se define como  
$\left[ f_{-a}(x) := f(a+x), \text{ \text{ para cada } } x \backslash \text{\text{real}}. \right]
```

Si f es una función real de variable real y a es un número, la función *trasladada* de f se define como

$$f_a(x) := f(a+x), \text{ para cada } x \text{ real.}$$

Inserción de texto en fórmulas

Como se observa en el ejemplo, al escribir el comando

`\text{Text}`

dentro del modo matemático, se procesa el argumento *Texto* prescindiendo del ambiente en el que está inmerso y se respetan los espacios, caracteres acentuados, etc. Los espacios situados antes

Comando	Sinónimo	Espacio	Comando	Sinónimo	Espacio
\thinspace	\,	\LARGE	\negthinspace	\!	\LARGE
\medspace	\:	\LARGE	\negmedspace		\LARGE
\thickspace	\;	\LARGE	\negthickspace		\LARGE

Cuadro 16.1: Comandos para espaciados horizontales

α	\alpha	β	\beta	γ	\gamma	δ	\delta
ϵ^*	\epsilon	ε	\varepsilon	ζ	\zeta	η	\eta
θ	\theta	ϑ	\vartheta	ι	\iota	κ	\kappa
λ	\lambda	μ	\mu	ν	\nu	ξ	\xi
π	\pi	ϖ	\varpi	ρ	\rho	ϱ^*	\varrho
σ	\sigma	ς	\varsigma	τ	\tau	υ	\upsilon
ϕ	\phi	φ	\varphi	χ	\chi	ψ	\psi
ω	\omega	\digamma	\digamma	\circ	\circ		
Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda
Ξ	\Xi	Π	\Pi	Σ	\Sigma	Υ	\Upsilon
Φ	\Phi	Ψ	\Psi	Ω	\Omega		

★ Cuando se tiene cargado el paquete `mathptmx` (versión 2/3/2003 PSNFSS-v9.0c), `\epsilon` produce la misma salida que `\varepsilon` y `\varrho` produce la misma salida que `\rho`. Las salidas ϵ y ϱ , en la tabla, son de las fuentes EC y aparecen más «delgadas» que el resto. Nosotros hemos utilizado la instrucción `\fontencoding{OML}\fontfamily{cmmi}\selectfont\symbol{N}` con $N=017$ para producir ϵ , y con $N=045$ para producir ϱ ; véase la sección II.1.4.6, donde se comentan los comandos utilizados más arriba para el uso puntual de fuentes.

Cuadro 16.2: Símbolos matemáticos: letras griegas

de la palabra `para` y después de `cada` son necesarios para separar la frase incluida dentro de `\text{ }` del resto de la fórmula. En modo matemático no se respetan los espacios en blanco pero sí funcionan todos los comandos de separación horizontal (introducidos en el apartado 7.3): por eso, el espacio antes de `real` en el ejemplo anterior ha podido ser introducido mediante un comando `_` usado fuera de `\text{real}`. El cuadro 16.1 contiene sinónimos de algunos de los comandos introducidos en la página 68 para producir separaciones horizontales, así como otros nuevos que están disponibles cuando se tiene cargado el paquete `amsmath`. Cuando este paquete está cargado todos estos comandos funcionan dentro y fuera del modo matemático.

§16.4 Letras griegas

LOS comandos para producir las letras griegas minúsculas consisten en la antibarra `\` unida al nombre de la letra (en inglés). Para las mayúsculas se pone la primera letra del nombre en mayúscula. El cuadro 16.2 recoge todos estos comandos.

Hay que tener en cuenta que las letras griegas no son letras de un tipo especial (como la itálica o la negrita) sino que son símbolos matemáticos y, en consecuencia, sólo pueden ser utilizadas en modo matemático (salvo el caso particular de la o-micron que se obtiene simplemente tecleando la letra ‘o’ en modo matemático).

EJEMPLO 16.15

```
\noindent {\bfseries Salida de S vídeo}\\\ 
Y (luminancia)\dotfill 1 Vp-p (75\,$\Omega$\$)\\ 
C (color)\dotfill 286 mVp-p (73\,$\Omega$\$)
```

Salida de S vídeo

Y (luminancia)	1 Vp-p (75 Ω)
C (color)	286 mVp-p (73 Ω)

Letras griegas; el comando \dotfill se comenta en la lección 18

§16.5 LATEX para químicos: el paquete chemsym

DESDE el punto de vista de la tipografía con LATEX, las ciencias químicas presentan dos tipos de dificultades. La primera es relativamente simple, pero molesta: las fórmulas empíricas contienen subíndices, las reacciones flechas, a menudo con texto sobre ellas, etc., características, todas ellas, propias del modo matemático de TEX. Sin embargo, no es posible identificar dicho modo como el adecuado para la escritura de fórmulas y reacciones químicas ya que, por ejemplo, este modo utiliza tipos especiales de carácter itálico, mientras que los símbolos de los elementos químicos son tradicionalmente tipografiados en tipos rectos.

Desarrollado por Mats Dahlgren [15], el paquete chemsym asegura una tipografía correcta de los símbolos de elementos químicos a través de una escritura cómoda del fichero fuente. El paquete chemsym define un comando para cada elemento de la tabla periódica (los 109 primeros): la barra de comando \ seguida del símbolo del elemento, de acuerdo con las recomendaciones de la IUPAC (International Union of Pure and Applied Chemistry) de 1997. Introduce, además, algunos comandos para símbolos usuales: \D para el deuterio, \Me, \Et, \Bu y \Pr para los grupos metil, etil, butil y propil, \OH, \COOH y \CH para OH, COOH y CH, respectivamente. El uso de estos comandos produce el símbolo de un elemento químico utilizando tipos rectos, independientemente de si se está utilizando el modo matemático o no; si a continuación no se escribe un subíndice o superíndice, paréntesis o corchete, se añade un pequeño espacio (algo menor que el proporcionado por \,).

Para escribir reacciones químicas necesitamos flechas: el comando

```
\longrightarrow
```

se utiliza para obtener una flecha larga hacia la derecha como ilustra el ejemplo 16.16; la forma de escribir distintos tipos de flechas se presentará en la segunda parte del libro (véase la sección II.5.3). En el ejemplo 16.17 que sigue utilizamos el comando

```
\xrightarrow[Deabajo]{Encima}
```

que está disponible en el paquete amsmath y que permite colocar la expresión *Deabajo* bajo la correspondiente flecha, y la expresión *Encima* encima de la misma.

EJEMPLO 16.16

Las reacciones de combustión se producen cuando un compuesto orgánico reacciona con oxígeno originando, en todos los casos, dióxido de carbono y agua. *par Ejemplo:*

```
\[
  \text{\textbackslash C\{4\}\textbackslash H\{10\}}+\text{\textbackslash frac\{13\}\{2\}\textbackslash O\{2\}}
  \text{\longrightarrow 4\textbackslash C\{0\}\{2\}+5\textbackslash H\{2\}\textbackslash O.}
\]
```

Fórmulas químicas con el paquete chemsym

EJEMPLO 16.17

La gasolina se obtiene a partir de fracciones pesadas del petróleo mediante una operación llamada *craqueo*, que consiste en romper mediante calor las cadenas largas de hidrocarburos. *par Ejemplo:*

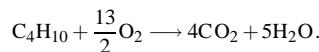
```
\[
  \text{\textbackslash C\{12\}\textbackslash H\{16\}}
  \text{\xrightarrow[Calor]{Catalizadores}}
  \text{\textbackslash C\{6\}\textbackslash H\{14\}+\textbackslash C\{6\}\textbackslash H\{12\}.}
\]
```

Reacciones químicas

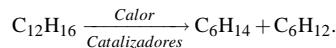
La coincidencia de algunos símbolos con nombres de comandos usuales hace necesaria la redefinición de éstos o el cambio de nombre para algunos símbolos; en chemsym se opta por la primera de las posibilidades. Así, el cuadro 16.3 muestra los nuevos nombres de los comandos afectados.

Las reacciones de combustión se producen cuando un compuesto orgánico reacciona con oxígeno originando, en todos los casos, dióxido de carbono y agua.

Ejemplo:



La gasolina se obtiene a partir de fracciones pesadas del petróleo mediante una operación llamada *craqueo*, que consiste en romper mediante calor las cadenas largas de hidrocarburos:



Comando original	Nuevo nombre con chemsym	Ejemplo
\H	\h	Acento en á
\O	\OO	Ø
\P	\PP	¶
\S	\Ss	§
\Re	\re	ℜ
\Pr	\pr	Pr

Cuadro 16.3: Nuevos nombres de algunos comandos en chemsym

Hemos de insistir en que los comandos definidos en el paquete chemsym funcionan dentro y fuera del modo matemático. Podemos así escribir $\$ \text{\textbackslash H_2\textbackslash O\$}$ o, más sencillo aún, $\text{\textbackslash H_2\textbackslash O\{}}$, para obtener H_2O . Esta última posibilidad de poder usar los caracteres _ y ^ de subíndices y exponentes fuera del modo matemático puede generar problemas de compatibilidad con otros paquetes: por ejemplo, con multicol y longtable (véase la sección II.3.3.3). Así mismo, si es utilizado con amstex o con rotating (véase la sección II.4.3) chemsym debe ser cargado después de cualquiera de ellos, mientras que con fancyhdr (véase la sección II.2.4.2) este último debe ser cargado con posterioridad.

El paquete define además el comando

`\kemtkn{Símbolo}`

que permite definir nuevos símbolos para ser tratados como símbolos químicos. Por ejemplo, podríamos definir

`\newcommand*\{\NH\}{\protect\kemtkn{NH}}`

A partir de esa definición podremos utilizar el comando `\NH` y el paquete añadirá un pequeño espacio en caso de que el carácter siguiente no sea un subíndice, exponente, paréntesis o corchete.

Con el paquete se distribuye el fichero de ejemplo `pertab.tex` cuya compilación produce una bella tabla periódica.

Ejercicios

16.1 Compile el fichero `ejercicio16.1.tex` y corrija los errores que se producen en la compilación. En el fichero `ejercicio16.1_sol.tex` puede encontrar comentada la solución correspondiente.

16.2 Escriba las siguientes fórmulas:

i) $x^3 - a^3 = (x - a)(x^2 + xa + a^2)$

ii) $x^{x+y^{x+y+z}}$

iii) $(AB)^{-1} = B^{-1}A^{-1}$

iv) $\sqrt{1 + \sqrt{1 + \sqrt{1 + x}}}$

v) $(a+b)^4 = \binom{4}{0}a^4 + \binom{4}{1}a^3b + \binom{4}{2}a^2b^2 + \binom{4}{3}ab^3 + \binom{4}{4}b^4.$

vi) Atención, la fórmula que sigue está en estilo `\displaymath`:

$$\text{div}(m) = \{n : n \text{ divide a } m\}.$$

PARA SABER MÁS

- Mejoras en las raíces, en II.5.5.9.
- Si necesita un control más preciso sobre el tamaño de las fórmulas consulte II.5.5.4.
- Para saber cómo evitar que la opción `spanish` de `babel` cambie automáticamente el punto, como separador decimal en fórmulas, por una coma, vea II.5.9.
- Hay muchos más símbolos matemáticos, operadores, funciones, flechas, etc., de los que usted ha aprendido en esta lección. Si necesita saber cómo se escriben estos símbolos, acuda al capítulo 5; pero no se alarme, no tiene que memorizarlos todos: como ya le dijimos en la lección 2, si usted está utilizando un programa como TEXNICCENTER, KILE, WINEDT o TEXTURES, sus entornos ya cuentan con botones y menús desde los que puede insertar muchos símbolos matemáticos.

- ▶ ¿Necesita utilizar delimitadores de tamaño variable (paréntesis, corchetes, llaves)? ¿Necesita escribir matrices o determinantes? Las herramientas para escribir este tipo de objetos las tiene en las secciones II.5.5 y II.5.6.
- ▶ A veces las fórmulas son muy largas. Como ya sabe, en modo matemático ordinario L^AT_EX parte las fórmulas, mientras que no lo hace en modo matemático resaltado. En la sección II.5.7 tiene todo lo que necesita para partir fórmulas por donde desee y escribirlas en varias líneas con una presentación a su gusto.
- ▶ Cuando necesite utilizar entornos propios para escribir matemáticas (lemas, teoremas, etc.) tendrá que aprender a crearlos, numerarlos, utilizarlos. Toda la información sobre estos entornos la tiene en II.5.12.
- ▶ El comando \text crea una caja y su argumento se procesa aislado del medio en el que está (por ejemplo, dentro del modo matemático). En la lección 18 puede aprender otras formas de producir cajas.

**OBJETIVOS**

- Asignar valores a los contadores y elegir el formato de su representación.
- Determinar y modificar el valor de una longitud.

CONTADORES y longitudes son objetos de uso constante en cualquier documento escrito con **LATEX**. Por ejemplo, en la lección 7 hemos visto las longitudes básicas de una página y el modo de incluir espacios verticales y horizontales; en las lecciones 6 y 11 han aparecido unidades numeradas como secciones o listas de ítems. Son sólo un botón de muestra de la profusa utilización que **LATEX** hace de estos objetos, aunque tal vez no haya consciente de ello. Contadores y longitudes están relacionados con números: en el caso de los contadores se trata sin más de números enteros ($0, 1, 2\dots, -1, -2\dots$); en el caso de las longitudes se trata de números enteros o decimales seguidos de una unidad ($5\text{ mm}, -2,6\text{ cm}$, etc.) pudiéndose utilizar en el código **LATEX**, indistintamente, el punto o la coma como separador decimal.

§17.1 Contadores

CADA contador tiene asociado un *nombre* que permite identificarlo. Dicho nombre suele ser el mismo del comando o elemento al que hace referencia, así: `page` es el contador que se utiliza para numerar las páginas; `chapter` es el contador que se utiliza para numerar los capítulos; `section` para las secciones, etc. En lo sucesivo nos referiremos genéricamente a dicho nombre como *NombreContador*.

No hay que confundir el *contador* con el *formato del contador*. El valor de un contador es siempre un número entero, mientras que su formato puede ser diverso: los capítulos en ocasiones se numeran como I, II, III...; los ítems de una lista pueden aparecer numerados con *a*, *b*, *c*, etc.

Están disponibles los siguientes formatos para los contadores con el resultado que se muestra al lado de cada uno de ellos:

<code>\arabic{NombreContador}</code>	1, 2, 3, 4...
<code>\alph{NombreContador}</code>	a, b, c, d...
<code>\Alph{NombreContador}</code>	A, B, C, D...
<code>\roman{NombreContador}</code>	I, II, III, IV...
<code>\Roman{NombreContador}</code>	I, II, III, IV...
<code>\fnsymbol{NombreContador}</code>	*, **, ***, ****, ...

En relación con esto conviene hacer algunas observaciones:

- Para los formatos `\alph` y `\Alph` el valor del contador no puede ser superior a 27, que es el número de letras disponibles en el abecedario español. En otro caso se produce un mensaje de error:
`Counter too long!`
- Para el formato `\roman` el resultado que aparece en la tabla anterior es el que se obtiene usando el paquete `babel` con la opción `spanish`. Sin `babel` (incluso con `babel` y otras opciones) el resultado obtenido es: i, ii, iii, etc.
- El formato `\fnsymbol` tiene su origen, como su nombre sugiere, en la costumbre, que algunos autores tienen, de utilizar símbolos para las marcas de las notas a pie de página, en lugar de números, cuando la cantidad de éstas es muy pequeña. De hecho, aunque este formato puede aplicarse a cualquier contador, nosotros nunca lo hemos usado salvo para notas a pie de página y con carácter muy excepcional. El resultado que hemos ilustrado más arriba es el que se obtiene usando el paquete `babel` con la opción `spanish`; en ese caso, el valor del contador no puede ser superior a 6. En nuestra opinión, si se van a incluir más de tres notas a pie de página, la utilización de este formato es estéticamente discutible. Si se utiliza el paquete `babel` con opciones diferentes de la opción `spanish` (y, desde luego, si no se utiliza dicho paquete) las marcas que se obtienen están en concordancia con la tradición anglosajona, que utiliza los símbolos*, †, ‡, etc.

Representación

Hemos distinguido entre contador y formato del contador; pero aún es necesario introducir un tercer elemento: la *representación del contador*. Asociado a cada contador `NombreContador` existe un comando, llamado representación del contador, que permite imprimir el valor del contador `NombreContador` en alguno de los formatos descritos anteriormente; este comando es:

`\theNombreContador`

Cuando `LATEX` define un contador la representación que inicialmente le asigna es la que corresponde al formato `\arabic`; si esta representación no es de su agrado puede cambiarla en cualquier momento y utilizar alguno de los otros los formatos disponibles, incluso puede incorporar a la representación otros caracteres y contadores previamente definidos. Todas estas redefiniciones se realizan a través de un comando `\renewcommand*`.

EJEMPLO 17.1

```
... El valor de esta página es \thepage, ...
Ahora cambiamos la representación
\renewcommand*{\thepage}{\roman{page}}
y el mismo comando proporciona \thepage, ...
\renewcommand*{\thepage}{\Roman{page}}
\thepage. \par
En la representación ... otros caracteres:
\renewcommand*{\thepage}%
{[C\thechapter P\arabic{page}]} \thepage.
```

Representaciones diferentes de un contador

Este ejemplo muestra cómo obtener el número de página en tres formatos diferentes. El valor de esta página es 141, en la representación original. Ahora cambiamos la representación y el mismo comando proporciona CXI y con otra CLXI

En la representación del contador pueden usarse caracteres y otros contadores: con sólo cambiar la representación del contador page ahora conseguimos incluir el número de lección y otros caracteres: [C17P141].

Gestión de valores

La gestión de los valores de un contador existente puede realizarse de dos maneras diferentes: asignando de forma directa un valor al mismo, o bien incrementando el valor inicial que tiene dicho contador. Dependiendo de lo que se quiera hacer, un procedimiento puede ser más adecuado que el otro. Los comandos que realizan esta tarea son los siguientes:

```
\setcounter{NombreContador}{Valor}
\addtocounter{NombreContador}{Valor}
```

\setcounter Asigna al contador *NombreContador* el valor entero *Valor*, con independencia del valor que tuviera anteriormente.

\addtocounter Incrementa el valor que en ese momento tiene el contador *NombreContador* con la cantidad *Valor*. El número entero *Valor* puede ser positivo o negativo. En el primer caso el nuevo valor del contador es mayor que el que tenía inicialmente, mientras que en el segundo el nuevo valor es menor que el valor inicial.

Las asignaciones que realizan los comandos anteriores tienen efectos a partir del lugar en el que son utilizados y su carácter es global, en el sentido de que aunque dichos comandos se introduzcan dentro de un grupo su campo de acción no queda restringido a dicho grupo. Este carácter global que tienen los contadores en L^AT_EX puede resultar incómodo en ocasiones; para esos casos conviene saber que el formato genuino de los contadores en T_EX, en el que está basado el de L^AT_EX, permite que el campo de acción de un contador quede restringido a un grupo. Para más detalles consulte el apartado PARA SABER MÁS al final de la lección.

EJEMPLO 17.2

```
Ésta es la lección \thechapter. Si le ...
\addtocounter{chapter}{2}
... pasaría a ser la número \thechapter.

¡Dejemos las cosas como estaban! Para ...
\addtocounter{chapter}{-2}
y el contador de lecciones vuelve a su
valor original \thechapter.
```

Ésta es la lección 17. Si le añadimos 2 unidades al valor actual del contador chapter, utilizado para numerar las lecciones, e imprimimos de nuevo la representación del contador, obtenemos que la lección pasaría a ser la número 19.

¡Dejemos las cosas como estaban! Para ello le añadimos -2 y el contador de lecciones vuelve a su valor original 17.

§17.2 Longitudes

AS longitudes que \LaTeX utiliza pueden tomar dos tipos de valores: valores *rígidos* y valores *elásticos*. Con valor rígido nos referimos a un valor preciso (1 cm, 7 mm...) que \LaTeX respeta. Un valor elástico consta de un *valor natural* y una cierta holgura positiva y/o negativa. Los valores de las longitudes pueden expresarse en las diferentes unidades que muestra el cuadro 17.1. También en este cuadro aparecen unidades de dos tipos: *unidades absolutas*, las nueve primeras, y *unidades relativas a la fuente en uso*, las tres últimas. Es fácil comprender que para determinados propósitos las unidades relativas son preferibles a las absolutas.

En la lección 7 han aparecido algunas longitudes, a propósito de los comandos `\enspace`, `\quad`, `\thinspace`, `\hoffset...`, cuyos valores son rígidos. Pero también han aparecido otras, en relación con los comandos `\bigskip`, `\medskip`, `\smallskip`, cuyos valores son elásticos. Por ejemplo, el comando `\bigskip` se define como

```
\vspace{12pt plus 4pt minus 4pt}
```

Lo que acabamos de decir sugiere, y de hecho así es, que dichas holguras se describen con ayuda de los prefijos `plus` (más) y `minus` (menos); por ejemplo, el código

```
\vspace{12pt plus 4pt minus 3pt}
```

indica a \LaTeX que debe introducir en ese lugar un espacio vertical cuyo valor natural es de 12pt, si bien es aceptable cualquier valor comprendido entre 9 pt y 16 pt. El valor que finalmente introduzca \LaTeX al componer el documento será el que convenga para que el espacio esté distribuido de forma homogénea y el resultado sea estéticamente satisfactorio, y ello dependerá del texto existente antes y después de dicho comando, por lo que modificar esos textos puede producir resultados diferentes en el valor final que tome el espacio vertical en cada caso.

Buena parte de las longitudes que \LaTeX utiliza toman valores elásticos para facilitar la tarea de composición del documento. Alguna de las holguras `plus` o `minus` puede estar ausente pero, si ambas aparecen, deben hacerlo en ese orden.

Los comandos `\bigskip`, `\medskip` y `\smallskip` corresponden a desplazamientos verticales (`\vspace`) de ciertas longitudes elásticas, cuyos valores respectivos están almacenados en:

<code>\bigskipamount</code>	<code>\medskipamount</code>	<code>\smallskipamount</code>
-----------------------------	-----------------------------	-------------------------------

Como en el caso de los contadores, cada longitud tiene asociado un nombre que la identifica, que indicaremos genéricamente con `\NombreLongitud`. Sus valores son números enteros o decimales (que pueden ser positivos o negativos) con una unidad de las indicadas en el cuadro 17.1. Obsérvese que, a diferencia de los contadores, tanto en la definición de longitudes como en su uso, el `\NombreLongitud` va precedido del carácter `\`.

Para obtener el valor de una longitud (véase el ejemplo 17.3) se emplea el comando

<code>\the\NombreLongitud</code>

que siempre la expresará en unidades pt, con el punto como separador decimal.

Desde el momento en que es creada en \TeX , a cada longitud se le asigna naturaleza de longitud rígida o elástica y sólo estas últimas admiten valores elásticos, mientras que los valores rígidos son

sp	La unidad más pequeña de \TeX (1 pt=65 536 sp, 1 mm= 186 712 sp)	
pt	Punto (1pt=0,351 mm)	
bp	Punto grande – ‘big point’ (1in=72 bp)	
dd	Punto Didôt (1dd=0,376 mm)	
mm	Milímetro (1mm=2,845 pt)	□
pc	Pica (1pc=12pt=4,218 mm)	□
cc	Cicero (1cc=12dd=4,531 mm)	□
cm	Centímetro (1cm=28,45 pt)	□
in	Pulgada (1in=25,4 mm=72,27 pt)	□
ex	Altura de una ‘x’	□
em	Anchura de una ‘M’, aproximadamente	□
mu	Unidad matemática (18 mu=1 em)	

Cuadro 17.1: Unidades de longitud válidas en \TeX

aceptados por todas las longitudes. Por ello, para asignar a una longitud un valor elástico (con los comandos que describimos en apartado siguiente) es necesario cerciorarse de la naturaleza de dicha longitud; un truco para conseguir esta información consiste en analizar el resultado producido por el comando `\the\NombreLongitud`.

Gestión de valores

La gestión de los valores de una longitud, ya sea asignándole un valor de forma directa o bien incrementando el valor que inicialmente tuviera, se realiza con los siguientes comandos:

```
\setlength{ \NombreLongitud}{Valor}
\addtolength{ \NombreLongitud}{Valor}
```

`\setlength` Asigna a la longitud `\NombreLongitud` un valor igual al argumento `Valor`, que debe ser una longitud válida para \TeX . Puede tratarse de un valor rígido, es decir, un número decimal seguido de una de las unidades que se especifican en el cuadro 17.1 (por ejemplo, 20 pt, 1 cm, etc.) o de un valor elástico (por ejemplo, 5mm plus 1mm minus .5mm), para el caso en el que a la longitud `\NombreLongitud` le puedan ser asignados valores elásticos. Una forma alternativa de asignar a la longitud `\NombreLongitud` un valor igual a `Valor` es `\NombreLongitud=Valor` (puede cambiarse el signo ‘=’ por un espacio o por nada). También puede asignarse valor a una longitud multiplicando por un factor otra longitud existente; en particular, el argumento `Valor` puede ser otra longitud (véase el ejemplo 17.3).

`\addtolength` Incrementa la longitud `\NombreLongitud` en una cantidad igual a `Valor`. Si la cantidad `Valor` es positiva, el nuevo valor de la longitud es mayor que el inicial, mientras que si la cantidad `Valor` es negativa, el nuevo valor de la longitud es menor que el inicial.

Con frecuencia el segundo comando resulta más cómodo que el primero debido a que, en muchas ocasiones, lo que se persigue es personalizar los parámetros que L^AT_EX tiene por defecto. En el apartado 7.2, ha aparecido un ejemplo de utilización de estos comandos a propósito de los comandos \textwidth y \hoffset.

Conviene llamar la atención sobre el hecho de que, a diferencia de lo que ocurre con los contadores, las asignaciones de longitud tienen un carácter local, lo que significa que si se realizan dentro de un grupo el efecto de tal asignación desaparece cuando el grupo finaliza, recuperándose a la salida del grupo el valor que tenía antes de iniciar el grupo.

EJEMPLO 17.3

```
La longitud que determina ...
el valor de dicha longitud es \the\parskip.
Vamos a fijar ahora un nuevo valor
... y en los sucesivos.
\setlength{\parskip}{7pt}

Debido a que los ejemplos ...
y su efecto desaparece cuando
salgamos de este entorno ejemplo.
```

Sobre el manejo de longitudes

La longitud que determina la separación entre párrafos se llama \parskip. En los ejemplos de este libro el valor de dicha longitud es 0.0pt. Vamos a fijar ahora un nuevo valor para dicha longitud que tendrá efecto en el nuevo párrafo y en los sucesivos.

Debido a que los ejemplos han sido construidos en este libro mediante un cierto entorno, esta modificación tiene carácter local y su efecto desaparece cuando salgamos de este entorno ejemplo.

Ejercicios

- 17.1** Como vimos en la lección 3, las longitudes que fijan la separación entre los párrafos y la sangría de los mismos son, respectivamente, \parskip y \parindent. Escriba un documento con varios párrafos y modifique el valor de dichas longitudes en diferentes párrafos utilizando los comandos descritos en esta lección.
- 17.2** En el archivo ejercicio17.2.pdf encontrará un documento con una única nota a pie de página y dos referencias a la misma. Utilice lo aprendido en esta lección y en la lección 13 para conseguir que la marca correspondiente al pie de página sea un asterisco (*) en vez de un número. En el archivo ejercicio17.2.tex encontrará una solución.

PARA SABER MÁS

- En las secciones II.3.2.1 y II.3.2.2 puede encontrar ejemplos de utilización de la representación de contadores en relación con la personalización de listas de ítems.
- A los contadores que L^AT_EX tiene definidos pueden añadirse otros. La sección II.1.5.1 describe el procedimiento para hacerlo. Se muestran allí también otros comandos de utilidad para el manejo de contadores, en particular, la forma de subordinar un contador a otro de manera que un incremento en una unidad en el valor del contador principal ponga automáticamente a cero el valor del contador subordinado.

- ▶ Al igual que ocurre con los contadores, pueden definirse nuevas longitudes. Los detalles puede encontrarlos en la sección II.1.5.2.
- ▶ En el cuadro 17.1 hemos incluido una lista de unidades de longitud. En la sección II.1.5.2 completaremos dicha lista con otras unidades de longitud utilizadas por L^AT_EX en relación con longitudes «infinitamente elásticas».
- ▶ En algunos casos resulta conveniente gestionar los contadores y las longitudes usando la sintaxis original de T_EX, que es más flexible que la de L^AT_EX. Los detalles puede encontrarlos en la sección II.8.2.

**OBJETIVOS**

- Crear cajas y enmarcarlas con varios estilos.
- Pequeñas páginas en medio del texto.
- Rayas o filetes.
- Mover cajas.
- Guardar y reutilizar cajas.
- Posicionamiento y distribución del espacio entre cajas.

PAQUETES COMENTADOS: fancybox

EN las lecciones anteriores hemos desarrollado las herramientas fundamentales de L^AT_EX. Con ellas podremos realizar la mayor parte de las estructuras que habitualmente aparecen en un documento, ya sea éste un artículo o un libro. Pero hay todavía algunas tareas usuales, como enmarcar objetos, pintar rayas horizontales y verticales o dibujar esquemas sencillos, para las que carecemos de los instrumentos necesarios.

Utilizando con un poco de astucia los entornos `tabular` es posible llevar a cabo algunas de esas tareas. Pero sólo algunas, porque resultados como los mostrados en los ejemplos 18.2 y 18.8 no pueden obtenerse con las herramientas que se han incluido en las lecciones anteriores.

Todas estas tareas están relacionadas con «cajas», como tendremos ocasión de comprobar. No podemos acabar esta primera parte sin referirnos, aunque sea de forma utilitarista, a las cajas, que constituyen el elemento primordial para el trabajo que realiza T_EX internamente y del que nos volveremos a ocupar, más ampliamente, en la segunda parte de este libro.

§18.1 Cajas y marcos

PARA T_EX cada carácter es una caja, la caja que contiene al carácter; cada línea es una caja, la caja que contiene a las diferentes palabras que constituyen la línea, y cada página es una caja, la caja compuesta por las cajas correspondientes a las diferentes líneas que forman parte de la página. Cada caja tiene un punto de referencia (que sirve para alinear las cajas a lo largo de

una línea base), una anchura (`\width`), una altura (`\height`) y una profundidad (`\depth`) —por ejemplo, la profundidad del carácter «p» es la longitud del «rabillo» que sobrepasa la línea base—. La suma de estas dos últimas longitudes está guardada en `\totalheight`.

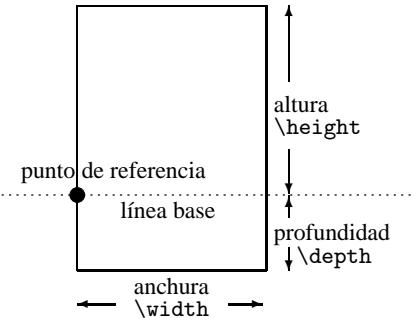
El usuario dispone de varios comandos para crear cajas.

<code>\mbox{Material}</code>	<code>\fbox{Material}</code>
<code>\makebox[Ancho] [Posición] {Material}</code>	<code>\framebox[Ancho] [Posición] {Material}</code>
	<code>\frame{Material}</code>

Los comandos `\mbox` y `\fbox` crean una caja que contiene al *Material*, y el segundo de ellos, además, pone un marco (frame) a dicha caja. Los comandos `\makebox` y `\framebox` son una extensión de los anteriores: producen cajas (sin marco y con marco, respectivamente) de anchura *Ancho*, colocando el *Material* en cierta *Posición* dentro de esa caja. Si no se emplean los argumentos optativos el resultado producido por los comandos de la segunda columna coincide con el obtenido con los correspondientes comandos de la primera columna. En todos los casos, y para obtener un resultado aceptable, la anchura del *Material* no debe sobrepasar el ancho de la línea. *Ancho* Puede ser cualquier longitud. Algunos anchos típicos son: `\width`, `\height`, `\depth`, `\totalheight`.

Posición Es un parámetro optativo que fija la posición que ocupará el *Material* dentro de la caja; puede tomar uno de los valores `l`, `r`, `c`, `s` que, respectivamente, corresponden a izquierda «left», derecha «right», centrado (valor por defecto), y «estirado a lo ancho» ‘stretched’. El parámetro `s` se puede utilizar cuando el argumento *Material* conste de varias palabras o de varias cajas. En ese caso LATEX intentará que dichos elementos se separen lo más posible, con la misma separación entre ellos, hasta agotar el *Ancho* de la caja. Para utilizar este parámetro, obviamente, es necesario indicar previamente el *Ancho*.

Las diferencias entre los comandos `\frame` y `\fbox` están en el punto de referencia de las cajas que ambos construyen, y en la separación del marco.



EJEMPLO 18.1

```
Colocamos la palabra \framebox[2\width]{hola}
en el centro ... Y ahora colocamos
\framebox[2\width][r]{hola} ...
Para la tercera posibilidad necesitamos ...
\framebox[2\width][s]{se separan}. Esta...
\fbox{caja} es diferente de esta otra
\frame{caja}.
```



```
Colocamos la palabra hola en el centro de un marco de
ancho el doble del ancho de dicha palabra. Y ahora colocamos
hola en el mismo marco pero a la derecha, en lugar de
centrada. Para la tercera posibilidad necesitamos al menos dos
palabras que se separan. Esta caja es diferente de
esta otra caja
```

El grosor de la raya con la que se dibuja el marco de la caja y la separación entre dicha raya y el objeto que enmarca están determinados por las longitudes

`\fboxrule` `\fboxsep`

cuyos valores por defecto son 0,4 pt y 3 pt, respectivamente.

Marcos llamativos: el paquete `fancybox`

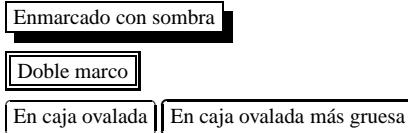
Las cajas ovaladas que delimitan los comandos y entornos en este libro han sido realizadas utilizando el paquete `fancybox`, desarrollado por Timothy Van Zandt [68], que introduce nuevos comandos, similares al comando `\fbox`, para enmarcar objetos.

<code>\shadowbox{Material}</code>	<code>\ovalbox{Material}</code>
<code>\doublebox{Material}</code>	<code>\ovalbox{Material}</code>

El funcionamiento de estos comandos es muy sencillo, como se aprecia en el ejemplo que sigue.

EJEMPLO 18.2

```
\shadowbox{Enmarcado con sombra}\par
\smallskip\doublebox{Doble marco}\par
\smallskip\ovalbox{En caja ovalada}
\Ovalbox{En caja ovalada más gruesa}
```



Objetos con marco y sombra

La longitud `\fboxsep`, introducida a propósito del comando `\fbox`, también actúa con estos comandos. La longitud `\fboxrule` controla el grosor de las rayas en los dos primeros comandos. Las rayas del comando `\doublebox` son de grosor diferente: la interior tiene un grosor de $0,75\fboxrule$ y la exterior de $1,25\fboxrule$; la separación entre ambas es $1,5\fboxrule$ plus 0,5pt. El grosor de las rayas en los comandos `\ovalbox` y `\Ovalbox` viene determinado, respectivamente, por las declaraciones `\thinlines` y `\thicklines` (véase el apartado PARA SABER MÁS del capítulo 4 en relación con el entorno `picture`).

`\shadowsize`

Es una longitud que determina el ancho de la sombra en `\shadowbox`. Su valor por defecto es 4 pt.

§18.2 Pequeñas páginas en medio del texto

AS cajas que hemos considerado en el apartado anterior son aptas para textos cortos (que no superen la anchura de la línea en curso) y su argumento no puede contener más de un párrafo (en otras palabras, no puede aparecer en el argumento una línea en blanco o el comando `\par`). Para textos con varias líneas, eventualmente con más de un párrafo, L^AT_EX dispone de dos herramientas para crear una caja, a modo de pequeña página dentro de la página ambiente, con un ancho que se establece de antemano.

`\parbox[Posición][Alto][PosRel]{Ancho}{Material}`

```
\begin{minipage}[Posición][Alto][PosRel]{Ancho}
Material
\end{minipage}
```

Aunque la versión comando puede servir para algunos propósitos con similares resultados, el entorno `minipage` funciona como una página auténtica en la que es posible incluir casi cualquier construcción estándar de L^AT_EX (las notas al margen constituyen una excepción) y por ello resulta, en general, más aconsejable. Estas herramientas se pueden utilizar para diversos propósitos, pudiendo aparecer en el argumento de muchos comandos y también constituir el *Objeto* de cualquier entorno, como `table`, `figure`, etc.

El significado de los argumentos es el siguiente:

Ancho Es un parámetro obligatorio que establece la anchura de la caja.

Material Es lo que se desea poner en la caja creada con `\parbox` o `minipage`. Habitualmente será un texto, pero puede ser cualquier otro elemento de L^AT_EX.

Posición Es un parámetro optativo para posicionar la caja con relación a la línea base. Puede tomar los valores `t`, `b`, `c` que corresponden, respectivamente, a que la parte superior de la caja (top), la parte inferior (bottom) o el centro, se sitúen en la línea base. Por defecto se utiliza la opción `c`.

Alto Es un parámetro optativo cuya inclusión exige que también se incluya la *Posición*. Si no se incluye, su valor es la altura que necesita la caja de anchura *Ancho* para incluir el contenido de *Material*. La longitud que corresponde a ese valor es `\height`. Pero puede hacerse una caja de cualquier altura, en cuyo caso la posición del *Material* dentro de esa caja sobredimensionada en altura está determinada por el parámetro *PosRel* que se describe a continuación.

PosRel Es un parámetro optativo cuya inclusión exige que los demás parámetros optativos se utilicen también; si no se incluye, toma el mismo valor que *Posición*. Su valor es una de las letras `t`, `b` o `c`, y sirve para fijar la posición relativa del *Material* dentro de una caja cuya altura (*Alto*) excede de la que se requiere para incluir dicho *Material*.

EJEMPLO 18.3

```
texto
\fbox{
\begin{minipage}[b][1.5\height][t]{.5\textwidth}
seguido de un texto metido en la caja
...
como si de una sola letra se tratara
\end{minipage}}
y más texto
```

seguido de un texto metido en la caja determinada por un entorno `minipage` que se comporta como si de una sola letra se tratara

texto y más texto

En `minipage` y `\parbox`, por defecto, no hay sangrado al comienzo de cada párrafo. Es decir, el valor de la longitud `\parindent` es `0pt`; pero se puede cambiar, dentro del entorno, con los comandos destinados a gestionar longitudes (véase la lección 17).

Notas a pie en el entorno `minipage`

El comando `\footnote` puede utilizarse dentro del entorno `minipage` (no así dentro de `\parbox`), pero su comportamiento dentro de este entorno resulta ligeramente diferente del ordinario: en primer lugar, las notas utilizan un contador distinto (`mpfootnote`) y el formato también es distinto (`\alph{mpfootnote}`) y, en segundo lugar, las notas se ubican en el borde inferior de la caja que contiene dicha `minipage` en lugar de hacerlo en el borde inferior de la página.

EJEMPLO 18.4

```
\begin{minipage}{.5\textwidth}
Pueden incluirse notas ordinarias%
\footnote{Como ésta.}
que aparecen al pie del
entorno. Y también otras\footnotemark
que aparecen al pie de la página.
\end{minipage}
\footnotetext{El texto de la que figura al
pie de la página.}
```

Pueden incluirse notas ordinarias^a que aparecen al pie del entorno. Y también otras¹ que aparecen al pie de la página.

^aComo ésta.

¹El texto de la que figura al pie de la página.

Notas a pie en el entorno `minipage`; a la derecha se ha simulado una página, pero el código de la izquierda corresponde únicamente a la parte legible

Estas notas dentro del entorno `minipage` pueden ser simultaneadas con notas a pie ordinarias, para ello:

- dentro del entorno `minipage`, en el lugar deseado, se escribe `\footnotemark`;
- después de salir del entorno `minipage` se escribe el *Texto* de la nota a pie, mediante el comando `\footnotetext{Texto}`.

§18.3 Rayas

AS rayas o filetes que aparecen en este libro, de cualquier grosor y con cualquier longitud, pueden considerarse cajas llenas de tinta (cajas negras). L^AT_EX puede utilizar para imprimirlas la siguiente sintaxis:

```
\rule[Elevación]{Ancho}{Alto}
```

donde *Ancho* indica la anchura de la caja y *Alto* su altura; *Elevación* es una longitud que determina el desplazamiento de la raya con relación a la línea base: cuando el valor es positivo la desplaza hacia arriba, y si es negativo lo hace hacia abajo.

EJEMPLO 18.5

```
Una raya gruesa \rule{1cm}{2mm} que sube
\rule[1mm]{1cm}{2mm}. \par Una raya normal
\rule{0.5cm}{0.4pt} que se hunde
\rule[-1mm]{0.5cm}{1.4pt}.
```

Una raya gruesa _____ que sube _____.
Una raya normal _____ que se hunde _____.

Rayas de distintos grosores que suben y bajan

EJEMPLO 18.6

```
La raya \makebox[0pt][1]{\rule[2.5pt]{1cm}{1pt}}%
\rule[1cm]{1pt} se llama caña en tipografía.\par
Y ésta \makebox[0pt][1]{\rule[4.5pt]{1cm}{1pt}}%
\rule[1cm]{3pt} se llama media caña.
```

La raya  se llama caña en tipografía.
Y ésta  se llama media caña.

Observe cómo se consigue superponer dos rayas metiendo la primera de ellas en una caja de anchura 0 pt

§18.4 Mover y reutilizar cajas

EN este apartado aprenderemos el modo en el que pueden moverse las cajas que hemos introducido a lo largo de la lección. Asimismo aprenderemos a «guardar cajas» para luego poder reutilizarlas, a modo de tampón, en otro lugar. Recordemos que las cajas que hemos considerado en esta lección corresponden a uno de los tipos siguientes:

ID (izquierda-derecha): Estas cajas se escriben de izquierda a derecha. Esta familia está compuesta por cuatro tipos de cajas: `\mbox`, `\makebox`, `\fbox` y `\framebox`.

Par (párrafos): Estas cajas contienen (o pueden contener) varias líneas y se escriben como si fuese un párrafo. Existen dos maneras de construirlas: `\parbox` y `minipage`.

Rayas: Las rayas o filetes son cajas negras creadas con el comando `\rule`.

Mover cajas verticalmente

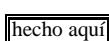
El comando `\hspace` permite mover cajas en sentido horizontal dentro de la línea, pero en ocasiones queremos que L^AT_EX escriba un determinado texto un poco más arriba o un poco más abajo de la línea base. Para solucionar este problema lo conveniente es colocar el texto (o en general, el material) dentro de una caja y entonces manipular dicha caja. L^AT_EX proporciona una función que hace esto por nosotros:

```
\raisebox{Elevación}[Alto][Prof]{Material}
```

donde *Material* representa lo que vamos a mover, *Alto* y *Prof* son, respectivamente, la altura y profundidad de la caja, en el caso de que se desee modificar los valores naturales que corresponden a *Material*. *Elevación* indica la longitud que se desplazará verticalmente la caja (hacia arriba si es positiva y hacia abajo si es negativa).

EJEMPLO 18.7

```
En ocasiones queremos
\raisebox{1ex}{levantar} un texto, y otras
veces lo queremos \raisebox{-1ex}{bajar}...
puede combinarse con otros,
como hemos \raisebox{-2mm}{\fbox{hecho aquí}}
\raisebox{-2mm}{[10pt][5pt]\fbox{y aquí.}}
```

En ocasiones queremos levantar un texto, y otras veces lo queremos bajar. El comando `\raisebox` puede combinarse con otros, como hemos  

Subir y bajar cajas (la caja más externa no aparece en el código)

Guardar y reutilizar cajas

A veces es más conveniente escribir el texto una sola vez y guardararlo en una caja, para poder utilizarlo posteriormente cuantas veces deseemos. Para realizar estas operaciones, L^AT_EX pone a nuestra disposición el siguiente comando:

```
\newsavebox{\NomCaja}
```

Este comando sirve para declarar la variable `\NomCaja` como perteneciente a una familia especial: aquella destinada a almacenar cajas. Posiblemente se esté preguntando: ¿por qué hay que definir previamente los comandos que van a servir para almacenar cajas, antes de definir su contenido, y no se utiliza el procedimiento usual en el que, simultáneamente, se crea el comando y se almacena su contenido? La razón reside en el funcionamiento interno de T_EX.

Una vez que hemos declarado un comando `\NomCaja` para almacenar una caja, ya es posible definir su contenido. Esto se hace con cualquiera de los dos comandos siguientes:

```
\sbox{\NomCaja}{Material}           \savebox{\NomCaja}[Ancho][Posición]{Material}
```

donde los argumentos *Ancho*, *Posición* y *Material* tienen exactamente el mismo significado que en `\makebox` o `\framebox`.

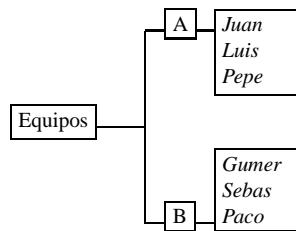
Para recuperar la caja que ha sido almacenada en `\NomCaja` debemos utilizar:

```
\usebox{\NomCaja}
```

Dicho comando imprime, en el lugar en el que se inserta, una copia de la caja almacenada en `\NomCaja`, dejando la caja intacta. Por esta razón, podemos usar su contenido tantas veces como deseemos.

EJEMPLO 18.8

```
\newsavebox{\EqA}\newsavebox{\EqB}%cajas auxiliares
\begin{lrbox}{\EqA}\begin{minipage}[t]{7ex}
\itshape\footnotesize Juan\,\, Luis\,\, Pepe
\end{minipage}\end{lrbox}
\begin{lrbox}{\EqB}\begin{minipage}[b]{7ex}
\itshape\footnotesize Gumer\,\, Sebas\,\, Paco
\end{minipage}\end{lrbox}
\fbox{Equipos}%
\rule{5ex}{0.4pt}\rule[-10ex]{0.4pt}{20ex}%
\makebox[0pt][l]{\rule[10ex]{2ex}{0.4pt}}\rule[-10ex]{2ex}{0.4pt}%
\makebox[0pt][l]{\raisebox{10ex}{\framebox{A}}}%
\raisebox{-10ex}{\framebox{B}}%
\makebox[0pt][l]{\rule[10ex]{2ex}{0.4pt}}\rule[-10ex]{2ex}{0.4pt}%
\makebox[0pt][l]{\raisebox{10ex}{\framebox{\usebox{\EqA}}}}%
\raisebox{-10ex}{\framebox{\usebox{\EqB}}}
```



Utilización de cajas en la realización de un esquema (véase el ejercicio 18.3). Los ingredientes básicos son entornos `minipage` y cajas de anchura 0 pt, para evitar el avance del «cursor»

¿Y si lo que queremos guardar es una caja del tipo `minipage`? L^AT_EX proporciona un entorno para este caso, cuya sintaxis general es

```
\begin{lrbox}{\NomCaja}
Material
\end{lrbox}
```

donde `\NomCaja`, comando preparado para almacenar una caja que ha sido previamente definida con `\newsavebox`, almacenará *Material*. Esencialmente, este entorno es una extensión del comando `\sbox` y para imprimir su contenido debemos usar también el comando `\usebox`. Los ejemplos 18.8 y 18.9 ilustran las cuestiones tratadas en este apartado.

EJEMPLO 18.9

```
\newsavebox{\rlist} \newlength{\rlistlength}
\newenvironment{rlista}[1][\columnwidth]{%
\setlength{\rlistlength}{#1}
\addtolength{\rlistlength}{-2\fboxsep}
\addtolength{\rlistlength}{-2\fboxrule}
\begin{lrbox}{\rlist}
\begin{minipage}{\rlistlength}
\begin{itemize}%
\end{itemize}\end{minipage}\end{lrbox}
\noindent\fbox{\usebox{\rlist}}\% FIN DEF.
\begin{rlista}
\item Merece la pena estudiar...
\item En particular, la definición...
\end{rlista}
```

Guarda y reutilizar el contenido de una caja

En este ejemplo se define un entorno de usuario, de nombre `rlista`, que permite generar listas de ítems recuadradas, en cajas de una anchura fijada por el usuario, e ilustrar su funcionamiento.

En la primera parte del código está la definición del entorno, que utiliza `lrbox`, `minipage` e `itemize`; observe el control de la anchura del entorno `minipage` para tener en consideración la anchura de las rayas.

En la segunda parte se hace uso del entorno así construido.

- Merece la pena estudiar la definición anterior.
- En particular, la definición de `\rlistlength`.

§18.5 Rellenar los espacios entre cajas

En ocasiones es necesario separar dos cajas interponiendo entre ellas el mayor espacio posible; tal es el caso que se presenta, por ejemplo, cuando en una carta de empresa se desea colocar en la esquina superior izquierda de la misma el nombre y logotipo de la empresa (que podría ser el contenido de un entorno `minipage`) y en la esquina superior derecha el nombre, teléfono, etc., de quien la envía (que podría ser, a su vez, el contenido de otro entorno `minipage`). La utilización del comando `\hspace` para conseguir la separación adecuada en tal situación resulta poco satisfactoria, porque exige conocer el espacio máximo que puede ser insertado. Los comandos que siguen son útiles para llenar el espacio existente entre dos objetos y resolver cómodamente situaciones como la descrita.

<code>\hfill</code> <code>\dotfill</code>	<code>\rulefill</code> <code>\vfill</code>
--	---

`\hfill` Es un comando que, insertado entre dos objetos situados en una misma caja con una anchura fijada (por ejemplo, una línea), introduce espacio vacío entre ellos hasta conseguir que aparezcan separados entre sí lo máximo que permita la anchura de la caja. Puede ser utilizado repetidas veces entre parejas de objetos.

`\rulefill` Es un comando con el mismo propósito que `\hfill`, salvo que en lugar de dejar vacío el espacio entre los objetos lo rellena con una raya horizontal.

`\dotfill` Es un comando con el mismo propósito que `\hfill`, salvo que en lugar de dejar vacío el espacio entre los objetos lo rellena con una línea de puntos.

`\vfill` Es un comando con una funcionalidad similar al comando `\hfill`; se aplica a cajas con altura y profundidad fijadas (por ejemplo, una página o un entorno `minipage` en el que se ha utilizado el argumento optativo *Alto*). Insertado entre dos objetos apilados verticalmente en una misma caja, introduce espacio vacío entre ellos hasta conseguir que aparezcan separados verticalmente entre sí lo máximo que permita la altura de la caja en la que se ubican.

EJEMPLO 18.10

```
\newsavebox{\logo}
\savebox{\logo}{%
  \sffamily\itshape
  \scalebox{3}[2]{\Large{L}}%
  \hspace{-4.6ex}%
  \reflectbox{\scalebox{3}{\Large{P}}}%
  \hspace{-4.3ex}%
  \raisebox{.7ex}{\Large{e}}\hspace{2ex}
  \parbox[b]{13ex}{%
    \bfseries La\pera Ediciones}
  \usebox{\logo} \hfill
  \parbox[b]{25ex}{%
    \itshape León Delfino\pera
    Director de contratación\pera
    ldelfino@lapera.es}
  \par\vspace{2em}
  \begin{minipage}{20em}
    D. Pedro Aprendiz\pera Avda. de la Fantasía, 16\pera
    Ciudad Imaginaria
  \end{minipage}
  \end{minipage}
  \par\bigskip
  Estimado amigo: \par\bigskip
  Tomamos nota de la petición que ...
}
```



La
Pera
Ediciones

León Delfino
Director de contratación
ldelfino@lapera.es

D. Pedro Aprendiz
Avda. de la Fantasía, 16
Ciudad Imaginaria

Estimado amigo:

Tomamos nota de la petición que nos hace y le agradecemos la confianza que deposita en nuestra empresa.

Los servicios que nos solicita requieren para su realización un uso intensivo de las cajas. Permítame que le ayude a descubrir las más destacadas. En primer lugar creamos una especie de «tampón» con el logotipo, a fin de poderlo usar repetidas veces. En su construcción también utilizamos cajas, algunas de las cuales serán descritas en la segunda parte de este libro.

Otra caja la reservamos para el director de contratación. Y también usted, para no ser menos, tiene la suya. Espero no herir con ello su sensibilidad, pero es que para nosotros encarjonar es lo prioritario.

Ejercicios

- 18.1 Utilizando las herramientas de esta lección, ¿cómo obtendría un resultado como el siguiente?

$\text{\LaTeX}\text{_2}_{\mathcal{E}}$ is the consolidation step in a comprehensive reimplemention of the \LaTeX system. The next major release of \LaTeX will be $\text{\LaTeX}\text{_3}$, which will include a radical overhaul of the document designers' and package writers' interface to \LaTeX .

$\text{\LaTeX}\text{_2}_{\mathcal{E}}$ es la etapa de consolidación en una reelaboración amplia de \LaTeX . La próxima versión significativa de \LaTeX será $\text{\LaTeX}\text{_3}$, que incluirá una revisión radical del modo de diseñar los documentos y los paquetes para \LaTeX .

- 18.2 Este ejemplo usa todos los parámetros del entorno `minipage`. ¿Podría generarlo?

Según la revista OCU Compra Maestra (núm. 226), en un par de zapatillas deportivas que se venden a 100 euros:

- 38 euros van a parar a la marca
- 32,5 euros son para el vendedor
- 16 euros para impuestos
- 13,5 euros cuesta la fabricación, de los cuales el operario que las fabrica recibe 0,5 euros

- 18.3** Para no complicar más el código, en el ejemplo 18.8 las rayas aparecen siguiendo la línea base y no se han tenido en cuenta la altura y la profundidad de las cajas. El resultado obtenido adolece de una asimetría entre el equipo A y el equipo B. Realice las oportunas modificaciones en el código para mejorar el resultado estético.
- 18.4** Trate de reconstruir la simulación de la página en el ejemplo 18.4. Por si lo necesita, puede encontrar la solución en un archivo `ejercicio18.4.tex` incluido en el CD-ROM.
- 18.5** Los comando `\hfill` y `\vfill` tienen un funcionamiento similar. Utilizándolos trate de construir algo parecido a esto:

Centrarse en la vertical
manteniendo la horizontal

CLMPS 2003

PARA SABER MÁS

- Hemos aprendido a hacer marcos alrededor de un objeto. En la sección II.3.5.2 encontrará información sobre cómo construir marcos alrededor de la página entera.
- Una marca al agua es un cierto tipo de caja que se repite en todas las páginas en la misma posición. En la sección II.3.5.2 puede aprender cómo hacerlas.
- Utilizando los entornos `tabular` y `minipage`, junto con los comandos `\hfill`, `\vfill`, `\hspace`, `\vspace`, etc., es relativamente sencillo realizar composiciones de una página distribuyendo cajas que contengan objetos diferentes: texto, gráficos, tablas... Los entornos `picture` y `pspicture` (véase, en la segunda parte del libro, el apartado PARA SABER MÁS del capítulo 4 y la sección II.4.4) tienen una mayor flexibilidad al permitir posicionar objetos en cualquier punto de la página mediante un sistema de coordenadas.
- Para construcciones complejas puede ser necesario utilizar tipos de cajas más versátiles que los utilizados en esta lección. En la secciones II.3.5 y II.3.5.1 encontrará más detalles sobre tales cajas y sobre el papel crucial que las cajas desempeñan en el trabajo interno de `TEX`.

Parte II

Para ser un L^AT_EXperto

Fundamentos

ESTE capítulo contiene aspectos muy variados sobre cuestiones básicas de L^AT_EX. Es un complemento a las dieciocho lecciones de la primera parte de este libro que versa fundamentalmente sobre «elementos de texto»: blancos, líneas, párrafos y páginas y familias de tipos. Las temáticas fundamentales tratadas se corresponden con las distintas secciones en las que se divide el capítulo y son las siguientes: control de los blancos entre palabras, la formación de párrafos y la paginación; adaptación de L^AT_EX a los distintos idiomas, en particular, a los idiomas del Estado español; y manejo de varias familias de tipos, entre ellas las de formato PostScript. El capítulo concluye con una sección de gran importancia, que completa la lección 17, en la que se presentan los contadores y longitudes «en todo su esplendor». Conviene insistir en la importancia de estos últimos elementos, a los que recurriremos continuamente en el resto del libro; entre ellos merecen una mención explícita las longitudes elásticas, con las que se producen situaciones divertidas, como la del último ejemplo del capítulo, pero relativamente complicadas.

El capítulo puede ser completado con el estudio de la introducción de caracteres especiales, entre ellos los acentuados, de forma totalmente independiente de la plataforma, que se incluye en el CD-ROM que acompaña a este libro (véase el apartado PARA SABER MÁS al final del capítulo).

1.1. Detalles sobre espacios, líneas y páginas

CON la experiencia adquirida sobre la escritura de texto usual, después del trabajo concienzudo de las lecciones de la primera parte, parecería que queda poco por aprender sobre elementos tan básicos como blancos, líneas o páginas. En efecto, así es, a pesar de lo cual en esta sección introduciremos algunos detalles más que, sin ser exhaustivos, le darán un conocimiento más profundo. Todos ellos han sido excluidos de la primera parte de este libro para no complicar más el aprendizaje inicial, pero ahora ya no le resultarán difíciles.

1.1.1. Espacios y signos de puntuación

Sabemos que el espacio entre dos palabras tiene una cierta elasticidad: cuando T_EX termina de componer un párrafo, ha dividido siládicamente algunas palabras y ha estirado, o comprimido, los blancos entre las palabras de cada línea. Estas tareas las realiza para conseguir que la composición

de todas las líneas del párrafo sea lo más perfecta posible, en el sentido de que los blancos no sean demasiado grandes y que todos sean iguales. El algoritmo que se ocupa de ello es enormemente complicado (véase el capítulo 14 de [34] para conocer detalles sobre el mismo).

Sin embargo, observando con mucha atención, podríamos detectar unos espacios mayores que otros; en concreto, los espacios que siguen a algunos signos de puntuación son mayores que los normales entre palabras. \TeX introduce por propia iniciativa estos espacios mayores porque forman parte de la más cuidada tradición tipográfica anglosajona. Desgraciadamente la situación es totalmente diferente en la tradición española (y en otras latinas, como la francesa). Donald Knuth conocía estas diferencias de criterio y, con un enorme respeto hacia otros idiomas, proporcionó formas simples de elegir uno u otro comportamiento. Los comandos

`\frenchspacing`

`\nonfrenchspacing`

son los adecuados para dicha elección. El primero anula el comportamiento estándar de \TeX antes descrito, mientras que el segundo lo recupera. Naturalmente, es innecesario incluir el segundo si lo que se desea es que estos espacios mayores estén activados; pero la ventaja es que estos comandos se anulan mutuamente, de forma que podemos ir variando el comportamiento según las necesidades, por ejemplo, para escribir unos fragmentos en español y otros en inglés. Téngase en cuenta, además, que se trata de declaraciones cuyo efecto se limitará al grupo en el que se incluyan (véase el apartado I.10.1).

El tratamiento especial de los espacios que siguen a los signos de puntuación consiste no sólo en hacerlos mayores, sino que, en caso de ser estirados o comprimidos, lo harán en mayor proporción que los espacios normales entre palabras. Este tratamiento especial se lleva a cabo sobre aquellos espacios que siguen a todos los signos de puntuación, así como a los signos de entonación (?) y (!) y a las comillas inglesas dobles ("'), aunque esto puede ser alterado en algunas fuentes. Aunque se trate de detalles muy menores, quizás al lector le resulte curioso saber, por ejemplo, que el punto y los dos puntos poseen un espacio posterior mayor que el espacio normal, mientras que la coma y el punto y coma poseen un espacio igual a éste, aunque con mayor capacidad de estiramiento y menor de contracción, siendo éstas inferiores, a su vez, a las del punto y dos puntos. El caso especial del punto explica la necesidad de modificar los puntos suspensivos cuando escribimos en español, una circunstancia que ya introducimos en la página 36.

Cuando `\nonfrenchspacing` está activo podemos encontrarnos con situaciones particulares, en las que los espacios mayores no son adecuados. Por ejemplo, ocasionalmente pueden aparecer en mitad de una frase palabras que, a pesar de acabar con un punto, no finalizan una frase (el ejemplo más sencillo corresponde a la abreviatura ‘etc.’). En ese caso el aumento del espacio posterior puede eliminarse introduciendo `_` después del punto, ya que este espacio no se altera nunca. Este problema sólo ocurre con los puntos que siguen a minúscula, puesto que cuando el punto (o el signo de entonación) sigue a una mayúscula, \TeX supone que no se trata del final de una frase, sino de una abreviatura o inicial. ¿Qué ocurre entonces si uno de estos signos, que sigue a una mayúscula, termina la frase? En tal caso debemos forzar el tratamiento especial del espacio posterior, insertando el comando

`\@`

antes del signo de puntuación, como en el ejemplo siguiente.

EJEMPLO 1.1

```
\nonfrenchspacing
Supongo ... con RENFE. No, ¡ni hablar!\par
Supongo ... con RENFE\@. No, ¡ni hablar!
```

Supongo que viajarás con RENFE. No, ¡ni hablar!

Supongo que viajarás con RENFE. No, ¡ni hablar!



1.1.2. Influyendo en la formación de los párrafos

La construcción de los párrafos es una tarea complicada que TeX realiza magníficamente a través de un algoritmo complejo, al que ya nos hemos referido anteriormente. Además de las posibilidades de rotura de líneas ya estudiadas en el apartado I.3.1, como los comandos \\ y \\newline, o el carácter ~ para evitar la separación de palabras, disponemos de otras vías para influir en la formación de un párrafo, ya sea para recomendar que una línea sea finalizada en un cierto punto o para modificar el número de líneas del párrafo.

La primera de estas formas de influencia la proporcionan los comandos

`\linebreak [Número]`

`\nolinebreak [Número]`

que recomiendan (`\linebreak`) o desaconsejan (`\nolinebreak`) al compilador el inicio de una nueva línea en el lugar preciso en el que aparecen. El parámetro optativo *Número* puede tomar uno de los valores 0, 1, 2, 3 ó 4, y a mayor valor de éste, más enérgica es la recomendación. No incluir este argumento equivale a especificar el valor 4, que representa la obligatoriedad de romper la línea en esa posición, para el primer comando, y la prohibición de hacerlo, para el segundo. Es preciso tener en cuenta que los requisitos que TeX exige a un «párrafo bien formado» son muy fuertes, por lo que es posible que no siga nuestras sugerencias con cualquiera de los comandos anteriores, a no ser que utilicemos el valor 4 para *Número*. En este último caso, es muy probable que el compilador emita un mensaje de tipo Underfull (es decir, espacios demasiado estirados), para indicarnos el mal resultado final del párrafo.

Hay una diferencia importante entre `\linebreak`, con cualquier valor del argumento, y \\ o `\newline`: estos últimos iniciarán una nueva línea dejando sin justificar la actual, mientras que el nuevo `\linebreak` justifica la línea en curso (la explicación es que \\ y `\newline` introducen `\parfillskip` al final de la línea, véase 1.5.2).

Podemos intentar modificar el número total de líneas de un párrafo creado por el compilador mediante la sintaxis:

`\looseness=Número`

siendo *Número* un entero, positivo o negativo. Incluyendo esta declaración en un párrafo (lo más natural es hacerlo al inicio del mismo) sugerimos al compilador que construya el párrafo de forma que el número total de sus líneas aumente en tantas como indica el valor de *Número*; si *Número* es negativo lo que deseamos es reducir la cantidad total de líneas.

A menudo este recurso se emplea para influir en la paginación, puesto que puede cambiar la altura de algunos párrafos de una página, o para eliminar la última línea de un párrafo, cuando ésta resulta muy corta. También puede ser útil en la escritura a dos o más columnas, para conseguir un mejor balanceo entre la altura de las mismas. Naturalmente el compilador no siempre atenderá

nuestra petición, porque hacerlo podría suponer formar un párrafo que no respete sus exigentes medidas de calidad. Es recomendable, por ejemplo, incluir `\looseness=1` en párrafos cuya última línea no es muy corta. Por otra parte, casi siempre es posible aumentar en una línea un párrafo suficientemente largo, sin poner en cuestión su calidad. Si se introduce `\looseness=Número` su efecto se limita al párrafo en el que se ha incluido.

EJEMPLO 1.2

```
En párrafos de medida muy corta ...
con \texttt{\textbackslash looseness}.

\looseness=1
En párrafos de medida muy corta ...
con \texttt{\textbackslash looseness}.
```

En párrafos de medida muy corta (estrechos) es más fácil conseguir aumentar el número de líneas con `\looseness`.

En párrafos de medida muy corta (estrechos) es más fácil conseguir aumentar el número de líneas con `\looseness`.

Uso del comando `\looseness`; observe que el párrafo aumentado posee espacios entre palabras bastante grandes; esto se debe a la brevedad del mismo

En cuanto a la interlínea, en el apartado I.3.5 describíamos el comando `\baselinestretch`, mediante el cual podemos modificar con comodidad esta interlínea. Sea cual sea la forma de modificar la interlínea (véase, por ejemplo, el comando `\fontsize` en la sección 1.4.6), la longitud

`\baselineskip`

almacena la altura de la interlínea actual o, más precisamente, la distancia entre las líneas base de dos líneas consecutivas. Puede ser útil conocer esta longitud explícitamente o recurrir a ella; por ejemplo, el código

```
\newcommand*{\PagLarga}{\enlargethispage{\baselineskip}}
\newcommand*{\PagCorta}{\enlargethispage{-\baselineskip}}
```

define dos comandos capaces de alargar o reducir la altura de una página, de forma que prevemos que dicha página pueda albergar una línea más o menos, respectivamente.

Es posible, aunque no recomendable, modificar directamente esta longitud. En realidad se trata de una longitud elástica, (véase la sección 1.5.2), pero desde luego no se le debe asignar un valor elástico, pues eso provocaría una distancia variable entre líneas consecutivas, lo que produciría una variación en la «negrura» de los párrafos que no es en absoluto adecuada y que se apreciaría incluso aunque la elasticidad fuera ínfima. Sí puede ser adecuado modificar la longitud, así como asignarle valores elásticos, cuando estamos colocando objetos en una caja vertical (véase 3.5.1).

1.1.3. Más ayuda en la paginación

La anterior técnica de uso de `\baselineskip` es un ejemplo de aplicación del comando `\enlargethispage`, herramienta fundamental en la paginación de un documento, ya comentada en el apartado I.7.3. Aunque L^AT_EX realiza un excelente trabajo automático para seleccionar los puntos de salto de página, ningún automatismo puede suplantar al autor del manuscrito, puesto que la decisión final puede depender del propio contenido de las páginas. Es obvio que el recurso a estas herramientas debe ser aplazado hasta el momento final y definitivo, momento en el que el documento se considera absolutamente terminado, si tal momento existe realmente...

Existe un recurso de ayuda a la paginación que no tiene por qué ser relegado hasta ese momento final. Se trata de los comandos

`\pagebreak [Número]`

`\nopagebreak [Número]`

donde *Número* es uno de los números 0, 1, 2, 3 o 4. Como se observa, son análogos a `\linebreak` y `\nolinebreak`, pero referidos a la rotura de páginas. El comando `\pagebreak [Número]` aconseja con intensidad creciente de 0 a 4 que ése es un lugar adecuado para realizar un salto de página; `\nopagebreak [Número]` aconseja con intensidad creciente de 0 a 4 que ése es un lugar inadecuado para realizar un salto de página. Si el argumento *Número* no se incluye, se toma como su valor máximo, 4, y supone una obligación. Se observa así que, por ejemplo, `\nopagebreak` puede ser insertado, en cualquier momento de la escritura, para tratar de evitar la separación, en páginas diferentes, de dos elementos consecutivos; si la rotura de página automática está finalmente lejos de estos elementos, la introducción del comando no provocará daño alguno.

Cuando estos comandos se usan entre dos párrafos se aplican a ese punto. Cuando se utilizan en medio de un párrafo se aplican inmediatamente después de completar la línea en la que están ubicados. De forma natural deben utilizarse por parejas, ya que si se quiere evitar un salto de página en un cierto lugar del documento, habrá que sugerir a L^AT_EX otro lugar cercano en el que se pueda realizar el salto, en caso de ser necesario. A diferencia de `\newpage` o `\clearpage`, no inician un nuevo párrafo después de producir el salto de página.

Otra herramienta útil en el tratamiento de los saltos de página es la declaración

`\samepage`

`\begin{samepage}`
Texto
`\end{samepage}`

Si incluimos en un grupo la declaración `\samepage` seguida de un texto, L^AT_EX intentará mantener todo ese texto en la misma página, pasando todo a la siguiente si no cupiera en la actual. En algunas referencias sobre L^AT_EX, por ejemplo en [35], se afirma que esta declaración nunca ha funcionado correctamente. Nuestra experiencia es que al menos en algunas ocasiones cumple su cometido y resulta práctica. Como se observa, existe en forma de entorno, cuya sintaxis viene a ser equivalente a `{\samepage Texto}`.

Un problema de naturaleza análoga a la formación de párrafos se plantea con la justificación vertical de las diferentes páginas. Si las páginas sólo contuvieran «texto continuo», en un único tamaño de letra, todas las páginas tendrían el mismo número de líneas y estarían perfectamente justificadas, es decir, imprimiendo dos páginas cualesquiera, las cajas de composición de ambas se podrían superponer de forma que coincidieran perfectamente, porque tendrían la misma altura. Pero la situación real es muy distinta: el texto puede contener tipos de diferentes tamaños, espacios verticales, gráficos, etc. En consecuencia, conseguir la justificación vertical de las páginas plantea problemas y, dado que la altura de la página no puede superar una longitud fijada de antemano (salvo que se utilice el comando `\enlargethispage`), sólo caben dos opciones: o bien poner el espacio sobrante al final de la página, dejando que la página sea «corta», o bien intentar repartir el espacio sobrante entre los diferentes párrafos y objetos. L^AT_EX dispone de sendas declaraciones, para incluir en el preámbulo, que determinan el tipo de solución que se adopta:

\raggedbottom	\flushbottom
---------------	--------------

que corresponden, respectivamente, con las soluciones antes consideradas. La segunda de ellas procede, naturalmente, estirando proporcionalmente todos los espacios verticales que son elásticos (la mayor parte de ellos lo son: `\parskip`, `\bigskip`, etc.). No es obligatorio elegir una de las dos declaraciones, puesto que cada clase de documento fija una de ellas. Así, `book` utiliza la segunda y `article` la primera, aunque estas determinaciones son alteradas por las opciones `twoside|oneside`, la última de las cuales siempre ejecuta la declaración `\raggedbottom`. Las páginas terminadas prematuramente con `\newpage` nunca son justificadas verticalmente, aunque `\flushbottom` esté activa; sí son ajustadas las que terminan por causa de un `\pagebreak`.

1.2. Casi todo sobre babel

El paquete `babel`, creado por Johannes Braams [6], es más que un paquete; podríamos denominarlo un «paquete marco» cuyo objetivo es facilitar la escritura de documentos `LATEX` que contienen texto en varios idiomas, así como la adaptación de un documento a las normas ortográficas y/o tipográficas de cada idioma. Esta adaptación se realiza, desde `babel`, ya sea mediante la introducción de nuevos comandos, según el idioma, o mediante cambios en el comportamiento estándar de otros.

En principio, la adaptación al idioma es siempre deseable pero, debido a los citados cambios, puede resultar, en algunas ocasiones, sorprendente y, a veces, problemática o incluso desagradable. El usuario de `babel` debe conocer, al menos someramente, a qué reglas responden los cambios sobre el comportamiento esperado, si estos cambios pueden ser anulados y, por supuesto, si los desea en su documento. Naturalmente, las reglas ortográficas no suelen ser cuestionadas; pero no ocurre lo mismo con las tipográficas, que, más que reglas, son «tradiciones» sobre las que los expertos no siempre logran tomar una postura unánime. En la primera parte de este libro ya hemos observado algunos de estos cambios producidos por `babel` o más precisamente, por su opción `spanish`, que prepara a `LATEX` para la escritura en español; recordemos, por ejemplo, el espacio pequeño anterior al signo porcentual (véase el ejemplo I.4.1) o el cambio del punto a la coma, como separador decimal en modo matemático (véase el apartado I.16.2).

El paquete `babel` se debe cargar en el preámbulo del documento con las opciones correspondientes a los distintos idiomas, que pueden ser tantos como se desee. Como ya sabemos, el comando `\documentclass` «transmite» cada una de sus opciones a los paquetes cargados que las entienden; por tanto, las opciones de `babel` pueden ser cargadas tanto en `\documentclass` como en el propio `\usepackage`. Todos los idiomas declarados mediante dichas opciones estarán disponibles en el documento. Sin embargo, es muy importante destacar el concepto de *idioma principal* para `babel`. Éste es el idioma correspondiente a la última opción incluida, considerando las opciones ordenadas según su orden de aparición: primero en `\documentclass` y, después, en `\usepackage{babel}`. Este *idioma principal* será el que se active al inicio del cuerpo del documento, es decir, al ejecutar `\begin{document}`.

Los idiomas disponibles en el momento actual (versión 3.7h del 1/3/2003) y el nombre de la opción correspondiente son los indicados en el cuadro 1.1. Cada idioma está asociado a un archivo

cuyo nombre suele coincidir con el nombre inglés para el idioma y que lleva extensión `ldf` (por `language definition`); por ejemplo, la opción `spanish` carga el archivo de nombre `spanish.ldf`. Este archivo es el que realmente contiene las particularizaciones del idioma en cuestión.

¿De qué se ocupa `babel` cuando lo cargamos en nuestros documentos? Fundamentalmente las tareas que realiza `babel`, o más precisamente los idiomas especificados, son:

- selecciona el algoritmo de división silábica correspondiente al idioma¹;
- traduce al idioma en cuestión los nombres de las unidades o antetítulos (véase la sección 1.2.2) y la fecha proporcionada por `\today`;
- introduce «símbolos taquigráficos» o «atajos de teclado» (el término inglés es «shorthands», a veces también denominados «abreviaciones») que permiten, entre otras cosas, introducir símbolos específicos del idioma de forma rápida desde el teclado;
- el *idioma principal* puede establecer una serie de órdenes que afectan a todo el documento, incluyendo aquellos fragmentos en los que sea otro el idioma «activo», a fin de unificar el formato del documento (véanse los ejemplos 1.3 y 1.4);
- finalmente, puede modificar la definición de algunos comandos y/o introducir otros nuevos para adaptar ciertas tareas al idioma, denominados «comandos extras».

Todas las tareas anteriores dependen de cada idioma: un idioma principal no establece necesariamente un formato global, los símbolos taquigráficos cambian en la forma y en sus acciones, etc.

El idioma principal, en el sentido antes definido, se activa en el momento de ejecutar la sentencia `\begin{document}` a fin de que las acciones llevadas a cabo por `babel` no entren en conflicto con otros paquetes cargados en el documento. Una consecuencia es que no se pueden utilizar las características del idioma en el preámbulo, por ejemplo para definir comandos; algunos idiomas, sin embargo, prevén esta posibilidad y ofrecen medios cómodos para hacerla realidad.

En general, cada idioma cargado como una opción de `babel` intenta leer un archivo de nombre la opción del idioma y extensión `cfg`; por ejemplo, cargado el idioma `catalan` se intentará leer el archivo `catalan.cfg`. Si el archivo no existe, no ocurre nada. Así el usuario puede escribir archivos `cfg`, con los nombres apropiados, para insertar en cada uno de ellos las modificaciones del comportamiento estándar del idioma que desee.

Cuando se modifican las opciones de `babel` se puede producir un error sin importancia. Concretamente, cuando, una vez compilado el documento con un idioma cargado como opción de `babel`, dicho idioma es eliminado y se vuelve a compilar el documento, L^AT_EX emitirá, en esta segunda compilación, el siguiente mensaje de error (suponiendo que el idioma eliminado es `english`):

```
! Package babel Error: You haven't defined the language english yet.
```

```
See the babel package documentation for explanation.
```

```
Type H <return> for immediate help.
```

```
...
```

```
1.8 \select@language{english}
```

```
?
```

¹Este idioma debe estar incluido en el formato `latex.fmt`; consulte la documentación de su instalación.

Idioma	Opciones	Idioma	Opciones
Afrikáner	africaans	Griego	greek, polutonikogreek
Alemán	austrian, german, germanb, ngerman, naustrian	Hebreo	hebrew
Alto serbio	uppersorbian	Holandés	ducth
Bajo serbio	lowersorbian	Húngaro	magyar, hungarian
Bretón	breton	Indonesio bahasa	bahasa
Búlgaro	bulgarian	Inglés	english, USenglish, american, UKenglish, british, canadian
Catalán	catalan	Irlandés gaélico	irish
Checo	czech	Islandés	icelandic
Croata	croatian	Italiano	italian
Danés	danish	Latín	latin
Escocés gaélico	scottish	Noruego	norsk, nynorsk
Eslovaco	slovak	Polaco	polish
Esloveno	slovene	Portugués	portuges, portuguese, brazilian, brazil
Español	spanish	Rumano	romanian
Esperanto	esperanto	Ruso	russian
Estonio	estonian	Saamí	samin
Euskera	basque	Serbio	serbian
Finlandés	finnish	Sueco	swedish
Francés	french, francais, canadien, acadian	Turco	turkish
Galés	welsh	Ucraniano	ukranian
Gallego	galician		

Cuadro 1.1: Idiomas de babel y las opciones que los seleccionan. Cada una de las múltiples opciones para un idioma supone un tratamiento diferente en algunas situaciones puntuales (por ejemplo, USenglish y UKenglish presentan la fecha dada por \today de distinta forma)

Este error aparece porque para cada uno de los idiomas cargados se escribe una «señal» del mismo en el fichero auxiliar AUX, de forma que en una segunda compilación, cuando éste se lee, se detecta un idioma que ya no existe. El error se subsanará, sin daño alguno, en una tercera compilación.

1.2.1. Comandos básicos de babel

Los comandos descritos en este apartado son algunos de los que afectan a la selección del idioma activo en cada parte del documento; esto significa que, normalmente, nunca se tendrá la necesidad de utilizarlos en un documento que sólo contiene un idioma. Un documento en el que se han cargado varias opciones de idioma se inicia, como ya se ha afirmado, activando el idioma

principal. Cuando llega el momento en el que se necesita escribir un texto en otro de los idiomas cargados, debemos recurrir a la declaración

```
\selectlanguage{Idioma}
```

donde *Idioma* es, naturalmente, el idioma que deseamos activar, escrito exactamente en la forma de la opción de babel correspondiente. Por ejemplo, la sentencia `\selectlanguage{UKenglish}` no es equivalente a `\selectlanguage{USenglish}`, puesto que ambos «*Idiomas*» se diferencian en algunos detalles. A partir de esta declaración, *Idioma* será el idioma activo y el texto que le siga, hasta una eventual nueva declaración del mismo tipo, seguirá todas las características de dicho idioma: traducción de antetítulos y fecha, reglas de división silábica, comandos «extras», etcétera.

Una forma distinta de activar un idioma es mediante los entornos:

```
\begin{otherlanguage}{Idioma}
Texto
\end{otherlanguage}
```

```
\begin{otherlanguage*}{Idioma}
Texto
\end{otherlanguage*}
```

que compondrán el *Texto* adaptado al *Idioma* indicado en el argumento, dejando de ser éste el idioma activo al salir del entorno. No obstante, su comportamiento no es exactamente el mismo; la diferencia entre ambos es que mientras la primera versión activa el *Idioma* completamente, la versión con asterisco activará las reglas de división silábica y los comandos «extras» del *Idioma*, pero no traducirá al *Idioma* los antetítulos ni la fecha.

EJEMPLO 1.3

```
\documentclass[english]{article}
\usepackage[spanish]{babel}
\begin{document}
\begin{itemize}
\item El idioma principal en este documento es \textsf{spanish}. Hoy es \today.
\item Observe que las viñetas del entorno \texttt{itemize} son pequeños cuadrados; ésta es una acción global tomada por el idioma principal.
\end{itemize}
%
\begin{otherlanguage}{english}
\begin{itemize}
\item Items are still labelled with small squares, even if the active language is english.
\item Today is \today.
\end{itemize}

```

Dos idiomas con babel

- El idioma principal en este documento es spanish. Hoy es 27 de agosto de 2003.
- Observe que las viñetas del entorno `itemize` son pequeños cuadrados; ésta es una acción global tomada por el idioma principal.
- Items are still labelled with small squares, even if the active language is english.
- Today is 27th August 2003.

Para activar un idioma de forma que afecte a un texto breve disponemos del comando:

```
\foreignlanguage{Idioma}{Texto}
```

Con este comando, en el momento de componer el *Texto*, el *Idioma* se activa parcialmente, en el sentido considerado anteriormente para `otherlanguage*`.

EJEMPLO 1.4

```
\documentclass{article}
\usepackage[spanish,english]{babel}
\begin{document}
\begin{itemize}
\item Now, the main language is english.
\item Items are labelled with a “bullet”. Today is the 27th August 2003.
\end{itemize}
\begin{otherlanguage*}{spanish}
\begin{itemize}
\item El idioma activo aquí es spanish.
\item Los ítems no se alteran, porque english, como idioma principal, no cambia el formato general. Hoy es 27th August 2003.
\end{itemize}
\end{otherlanguage*}
```

Los mismos idiomas con babel, pero distinto «idioma principal»

Finalmente disponemos del entorno:

```
\begin{hyphenrules}{Idioma}
Texto
\end{hyphenrules}
```

que sólo activará en su cuerpo las reglas de división ortotipográfica correspondientes a *Idioma*.

Cuando los cambios de idioma activo son frecuentes o son varios los idiomas manejados en un documento, puede llegar un momento en que no estemos seguros de cuál es el idioma activo en un cierto punto del documento fuente; los siguientes comandos pueden ser útiles en tales casos:

<code>\languagename</code>	<code>\iflanguage{Idioma}{ParteV}{ParteF}</code>
----------------------------	--

El primero de ellos sólo imprime el nombre del *Idioma* activo (en realidad imprime el nombre de la opción de babel asociada al *Idioma*). El segundo es un condicional (para otros condicionales genéricos véase la sección 8.6) que actúa en la forma siguiente: si el idioma activo es el de nombre *Idioma*, se llevarán a cabo las acciones incluidas en el argumento *ParteV*; en caso contrario, las acciones que se ejecutarán serán las contenidas en *ParteF*.

1.2.2. La traducción de antetítulos

Los antetítulos de capítulos, figuras, índices, etc., se definen en los comandos que aparecen en el cuadro 1.2 y pueden ser modificados mediante el correspondiente `\renewcommand*`. Los distintos idiomas proporcionan una traducción para cada uno de ellos, de lo que el usuario se beneficia automáticamente al cargar el idioma de su interés en babel. En el cuadro 1.2 se muestran las traducciones a las distintas lenguas del Estado español. Puede ocurrir, sin embargo, que al usuario no le satisfagan las traducciones de dicho idioma y en tal caso cambiarlas supone una mayor complicación, aunque tan sólo ligera. Para modificar las traducciones de cualquiera de los idiomas cargados con babel disponemos de varias estrategias. La más inmediata sería utilizar la habitual, por ejemplo

```
\renewcommand*\contentsname{Contenidos}
```

pero esta forma presenta dos problemas: no funcionará si se declara en el preámbulo, ya que las definiciones del idioma principal se activan en el `\begin{document}`, y se desactivarán cuando se seleccione otro idioma (en caso de haber cargado más de uno en babel), ya que cada cambio de idioma selecciona las traducciones del nuevo idioma activado.

La mejor forma de proceder es utilizar los comandos que babel proporciona para ello, mediante la sintaxis siguiente:

```
\addto\captionsIdioma {Redefinición1 Redefinición2 ...}
```

A cada idioma le corresponde el comando `\captionsIdioma`, por ejemplo, `\captionscatalan` al catalán, que almacena todas las traducciones. Los argumentos indicados como *Redefinición1*, *Redefinición2*, etc., deben ser sustituidos por la redefinición del antetítulo que se desea cambiar. Por ejemplo, para llevar a cabo la traducción antes mostrada, escribiríamos

```
\addto\captionsspanish{\renewcommand*\{\contentsname\}{Contenidos}}
```

mientras que

```
\addto\captionsspanish{\renewcommand*\{\contentsname\}{Contenidos}%
    \renewcommand*\{\tablename\}{Tabla}}
```

cambiaría también el antetítulo de las leyendas de los entornos `table`.

Pero es precisa una precaución esencial: las redefiniciones anteriores deben hacerse cuando el *Idioma* del comando `\captionsIdioma` no esté activado. Lo ideal es ubicarlas en el preámbulo, donde ningún idioma está activo. Si es imprescindible realizar un cambio en el cuerpo del documento, entonces debemos asegurarnos de que otro idioma está activo (introduciendo un cambio artificial de idioma, si es preciso) antes las modificaciones y, después, volver al idioma deseado.

Algo similar a lo que ocurre con las traducciones anteriores se presenta con la fecha. Cada idioma posee un comando

```
\dateIdioma
```

que redefine, internamente, el comando `\today`. Si queremos redefinir éste de forma segura, lo mejor es hacer:

```
\addto\dateIdioma {\renewcommand*\{\today\}{Redefinición}}
```

si bien la redefinición del comando `\today` es, a menos que se le quiera asignar un valor explícito y fijo, técnicamente complicada (véase la página 478).

1.2.3. Comandos para taquígrafos

En la exposición general de babel hemos citado la introducción de «símbolos taquigráficos» como una de las tareas importantes que realizan los distintos idiomas del paquete. ¿En qué consisten exactamente estos métodos taquigráficos? Se trata de convertir ciertas combinaciones de caracteres en comandos; eso sí, comandos que no comienzan con el tan usual carácter `\`. Por ejemplo, en `spanish` se define un método taquigráfico como la combinación "a, mediante el cual el código 1" a proporciona 1.^a como salida. La idea de estos métodos taquigráficos es la de proporcionar formas rápidas de escritura de comandos complejos o de nombres largos, o bien aproximar

la forma de obtener algunos signos, por ejemplo letras acentuadas, a la forma habitual de escribirlos desde el teclado en cada idioma. Desde este punto de vista la idea puede ser atractiva, aunque, en algunos casos y teniendo en cuenta las facilidades que proporciona el paquete `inputenc`, su interés puede ser muy cuestionable, más aún si se tiene en cuenta que, en ocasiones, pueden provocar incompatibilidades con algunos paquetes (por ejemplo, en la sección 5.11 hemos tenido que desactivar << y >>, que son métodos taquigráficos para la opción `spanish`, por su incompatibilidad con el paquete `amscd`). Los métodos taquigráficos mantienen, eso sí, la independencia del sistema, lo que no es del todo cierto, por ejemplo, con las opciones del paquete `inputenc`.

En `babel` se distinguen tres niveles de métodos taquigráficos: el del sistema (es decir, el propio `babel`), el del idioma (uno para cada idioma cargado) y el del usuario. Si un método está definido en más de uno de esos grupos, la definición válida es la del primer grupo en el orden de prelación establecido por `babel`, que es: usuario, idioma, sistema.

Los siguientes comandos permiten definir métodos taquigráficos por el usuario (automáticamente incluidos en el grupo del usuario):

`\useshorthands{Car}`

`\defineshorthand{Cars}{Definición}`

El primero de ellos declara a su argumento *Car*, que debe ser un único carácter, como «activo», es decir, que jugará un papel especial como primer carácter de uno o más métodos taquigráficos. El segundo comando define un método identificado por su argumento *Cars* que debe consistir en uno o dos caracteres. El primero de los caracteres de *Cars* debe haber sido declarado como activo en un comando `\useshorthands`, o bien ser el primer carácter de algún método de un idioma declarado en el documento.

Los siguientes comandos permiten activar o desactivar temporalmente todos los métodos taquigráficos que comienzan por cualquiera de los caracteres incluidos en sus argumentos:

`\shorthandon{Car1 Car2...}`

`\shorthandoff{Car1 Car2...}`

Estos comandos pueden aplicarse a métodos de cualquiera de los tres niveles. Al activar un idioma, mediante los comandos apropiados expuestos en la sección 1.2.1, se activan sus métodos taquigráficos; por tanto, si anulamos un método que poseen dos idiomas mediante `\shorthandoff` y después activamos uno de esos idiomas, el método se reactivará. En general no disponemos de medios para desactivar los métodos de un idioma específico, aunque algunos idiomas sí proporcionan herramientas para ello (por ejemplo `spanish`, véase la página 176).

Finalmente, mediante el comando

`\languageshorthands{Idioma}`

se activan los métodos taquigráficos definidos en *Idioma*, independientemente de cuál sea el idioma activo en el momento de ejecutarlos y desactivando los métodos de dicho idioma activo.

1.3. Las lenguas del Estado español

EN esta sección abordaremos las particularidades introducidas por los idiomas `basque`, `catalan`, `galician` y `spanish` de `babel`. De los tres idiomas autonómicos presentamos exclusivamente sus traducciones de los antetítulos en el cuadro 1.2, y sus métodos taquigráficos. Estos idiomas de

Comando	Catalan (catalan)	Español (spanish)	Euskera (basque)	Gallego (galician)
\partname	Part	Parte	Atala	Parte
\chaptername	Capítol	Capítulo	Kapitulua	Capítulo
\appendixname	Apèndix	Apéndice	Eranskina	Apéndice
\abstractname	Resum	Resumen	Laburpena	Resumo
\contentsname	Índex	Índice general ^a	Gaien Aurkibidea	Índice Xeral
\listfigurename	Índex de figures	Índice de figuras	Irudien Zerrenda	Índice de Figuras
\listtablename	Índex de taules	Índice de cuadros	Taulen Zerrenda	Índice de Táboas
\bibname	Bibliografia	Bibliografía	Bibliografia	Bibliografía
\refname	Referències	Referencias	Erreferentziak	Referencias
\indexname	Índex alfàbetic	Índice alfabético	Kontzeptuen Aurkibidea	Índice de Materias
\glossaryname	Glossary ^b	Glosario	Glossary ^b	Glossary ^b
\figurename	Figura	Figura	Irudia	Figura
\tablename	Taula	Cuadro	Taula	Táboa
\proofname	Demostració	Demostración	Frogapena	Demostración
\pagename	Pàgina	Página	Orria	Páxina
\seename	Vegeu	Véase	Ikusi	véxase
\alsoname	Vegeu també	Véase también	Ikusi, halaber	véxase tamén

^aEn la clase article tan sólo ‘Índice’.

^bNo ha sido traducido.

Cuadro 1.2: Traducción de los antetítulos a los idiomas del Estado español

babel apenas realizan otras tareas. Por el contrario, el idioma spanish es el que introduce un mayor número de comandos y más altera el comportamiento estándar de LATEX, por lo que su exposición será más amplia.

1.3.1. Métodos taquigráficos y algunos comandos extras

A continuación se detalla cada uno de los atajos de teclado introducidos por los idiomas que nos ocupan en esta sección. En general, todos los métodos que se inicien con un apóstrofo (') sólo estarán activos si se carga el paquete babel con la opción activeacute. En el caso del catalán también existe la opción activegrave, como se indica en la descripción del método que se inicia con dicho acento.

Es preciso indicar un detalle a propósito de estas opciones que activan el apóstrofo o el acento grave como métodos abreviados. Si, por ejemplo, escribimos {\itshape la palabra ‘hola’} el compilador enviará un mensaje de error, porque interpreta la pareja ’} como un comando; para evitar este pequeño inconveniente, siempre que estén activadas las citadas opciones, debemos recordar escribir en su lugar {\itshape la palabra ‘hola’{} }.

Métodos taquigráficos en español

- 'a' [] 'A'** vocales ('e, etc.) con acento agudo, mayúsculas y minúsculas;
- 'n' [] 'N'** ñ y Ñ;
- "u" [] "U"** ü y Ü;
- "c" [] "C"** ç y Ç;
- "a" [] "A"** voladitas en la forma dada por \sptext (v. pág. 37); también existen "o y "O;
- "<" [] ">"** comillas latinas « »;
- << [] >>** \begin{quoting} \end{quoting} (véase la página 175); puesto que con la codificación T1 las combinaciones << y >> constituyen ligaduras que proporcionan los signos « y », se establece un conflicto con este método taquigráfico, conflicto que se resuelve en favor de este último método, a no ser que se desactive mediante \shorthandoff< o mediante los comandos específicos de desactivación de spanish (véase la página 176);
- "/" []** barra '/' movida ligeramente hacia abajo, ¿aprecia la diferencia entre 1/2 y 1/2 (generados, respectivamente, con 1/2 y 1"2)? Su uso tiene sentido solamente con algunas familias de fuentes, como times (véase la sección 1.4);
- "rr" [] "RR"** inserta rr o RR normalmente, a menos que la palabra sea dividida siládicamente justo antes de esta combinación, en cuyo caso sólo aparecerá una r o R en la siguiente línea; este comportamiento materializa una norma de la Real Academia Española;
- "-" []** análogo al comando \- pero permite que se divida en otros lugares la palabra en la que se introduce;
- "=" []** imprime un guión, permitiendo guiones de división silábica en otros lugares de la palabra, lo que no hace LATEX por defecto; aunque esto no es recomendado por las normas ortotipográficas, puede ser un recurso en anchos de línea muy pequeños;
- "~" []** introduce el denominado «guión estilístico»: el guión de división silábica aparece al final de la línea y al principio de la línea siguiente; se utiliza a veces en lingüística para indicar que un guión debería aparecer en el lugar, incluso aunque la palabra no hubiera sido dividida;
- ~- []** introduce un guión simple (-) evitando la división; también está disponible para los otros guiones (~-- y ~---); con este método se evita, por ejemplo, que la raya de abrir un inciso sea separada de la primera palabra del mismo.

Métodos taquigráficos y comandos en catalán

- 'a'** acento agudo sobre la letra 'a', permite la división silábica en cualquier posición de la palabra; naturalmente existen métodos análogos para el resto de las vocales, tanto minúsculas como mayúsculas: 'A, 'e, 'E, 'i, etc.;
- 'a'** acento grave sobre la letra 'a'; los mismos comentarios que en el caso anterior se aplican a éste; la opción de babel que activa este método es activegrave;
- "i"** i con diéresis, permite división silábica en otras partes de la palabra;
- "c" [] "C"** cedilla (ç), minúscula y mayúscula;

- "1** **"L** 1 geminada, minúscula y mayúscula, como en ‘íllustre’;
- "<** **">** comillas latinas « »;
- "-** como en español;
- "|** elimina la posible ligadura entre dos caracteres en el lugar en el que se inserta; por ejemplo, ‘finestra’ es el resultado de f" | inestra, mientras que ‘finestra’ es el de finestra.

El idioma catalán introduce algunas definiciones o redefiniciones propias de comandos:

<code>\l.1 \lgem</code>	<code>\up{Voladitas}</code>	<code>\-</code>
<code>\L.L \Lgem</code>		

Los comandos `\l.1` y `\lgem` son sinónimos del método taquigráfico "1 y algo similar ocurre con las versiones en mayúscula. `\up` es sinónimo de `\textsuperscript` (véase página 37). Finalmente, `\-` es una redefinición del mismo comando, descrito en la página 24, que lo hace idéntico al método "-" (aquí se puede observar el discreto interés de algunos métodos taquigráficos: no es mucho más difícil teclear `\-` que `"-`).

Métodos taquigráficos en euskera

- "n** **"N** ñ y Ñ;
- "<** **">** comillas latinas « »;
- "|** elimina la ligadura en el lugar en que se incluye;
- "-** como en los anteriores idiomas.

El comando `\-` se redefine en basque de la misma forma que en catalán.

Métodos taquigráficos en gallego

- 'a** **'A** á, Á; existe también para las otras vocales;
- "n** **"N** ñ, también están disponibles en mayúsculas;
- "u** **"U** ü, Ú;
- "o** **"a** ordinales masculino y femenino; la salida que producen es distinta a la de `spanish`; por ejemplo, 1^a es la salida que proporciona 1" a;
- "|** como en los anteriores idiomas;
- "-** como en los anteriores idiomas.

También en gallego se redefine `\-`, en el mismo sentido que en catalán y euskera.

1.3.2. Comandos y acciones de la opción `spanish`

La opción `spanish` de `babel`, la adecuada para la escritura de textos en español, fue iniciada por Julio Sánchez y es mantenida actualmente por Javier Bezos [3]. Además de las traducciones y los métodos taquigráficos, realiza varias tareas y provee al usuario de numerosas posibilidades de configuración y variaciones del comportamiento estándar; todo ello se describe a continuación.

Entre todas las posibilidades de `spanish` algunas se refieren a la escritura de fórmulas; éstas serán abordadas en la sección 5.9.

Cuando `spanish` es el idioma principal

En esta situación el idioma `spanish` ejecuta, en el momento de leer `\begin{document}`, un grupo de comandos; todos ellos se «incluyen» en el comando

```
\layoutspanish
```

¿Qué hace este grupo de comandos? Pues bien, lleva a cabo varias tareas, decididas por el diseñador de la opción con el objetivo de dar uniformidad a todo el documento. Concretamente se determinan los siguientes elementos:

- la letra ‘ñ’ se incluye en las representaciones `\alph` y `\Alph` de los contadores;
- La representación `\fnsymbol` para el contador de notas a pie de página toma los valores *, **, etc. (v. pág. 108);
- la representación `\roman` se redefine para que imprima los números en tipos versalita y no en minúsculas (v. pág. 140);
- los números de las secciones y unidades de nivel inferior van seguidos de punto, tanto en el título como en las entradas en el índice general;
- las etiquetas de los ítems en los entornos `itemize` y `enumerate` se alteran: en `enumerate` se opta por el esquema 1., a), 1), a'), para los cuatro niveles y en `itemize` el esquema elegido es ■, •, ◦, ◇. Estas etiquetas pueden ser cambiadas por el usuario, véase [3, pág.12].

Observe que algunos de estos comportamientos han sido tomados como los naturales en las lecciones de la primera parte; esto se debe, naturalmente, al preámbulo estándar que definimos en la lección 2, que declaraba, como es ahora obvio, el español como idioma principal.

Una vez lanzado este grupo no puede ser, en principio, desactivado, aunque sí se puede evitar que sea ejecutado (véase el apartado sobre desactivación). El objetivo de esto es que no se cambie el aspecto de los elementos anteriores a lo largo del documento, independientemente del idioma en que estén escritas algunas partes del mismo, y así se consiga una uniformidad de diseño en todo el documento.

Aunque sea el idioma principal, los distintos elementos de un idioma no están disponibles en el preámbulo del documento (como ya se ha dicho, los comandos y acciones de las opciones de `babel` son activadas con `\begin{document}`). Por tanto, no podemos utilizar traducciones, métodos taquigráficos o comandos extras en el preámbulo, lo que podría resultar útil en comandos como `\title` o semejantes. La opción `spanish` proporciona el comando:

```
\selectspanish
```

cuyo objetivo es poder activar el idioma español en el preámbulo. Incluyendo este comando en el preámbulo todo lo correspondiente a `spanish` se activa en el mismo; se debe prestar atención, no obstante, a posibles incompatibilidades con otros paquetes, por lo que es recomendable incluir `\selectspanish` después de cargar todos los paquetes.

Comandos extras de spanish

Algunos de los nuevos comandos que introduce spanish, o de las redefiniciones de comandos de L^AT_EX, ya han sido presentados; por ejemplo: \dots, \sptext y \% (véase la lección 4). A ellos hay que añadir el comando

```
\lsc{Texto}
```

que imprime *Texto* en versalitas minúsculas (independientemente de si *Texto* contiene mayúsculas y minúsculas) y que [3] recomienda para las siglas. Además, spanish ejecuta el comando \frenchspacing que controla el espacio después de los signos de puntuación (consúltese la sección 1.1.1).

Todos los comandos anteriores están «guardados» en un macrocomando:

```
\textspanish
```

de forma similar al caso de \layoutspanish.

Existen otros dos grupos de macrocomandos en spanish:

```
\shorthandsspanish
```

```
\mathspanish
```

El primero contiene, y activa, todos los métodos taquigráficos, mientras que el segundo hace lo correspondiente con varios elementos propios de las fórmulas y se describe en la sección 5.9.

Disponemos también del entorno quoting que no es un «extra» realmente, en el sentido de que no forma parte de ninguno de los grupos de comandos anteriores:

```
\begin{quoting}
Texto
\end{quoting}
```

Recuerde que << y >> son atajos equivalentes a \begin{quoting} y \end{quoting}.

EJEMPLO 1.5

```
\begin{quoting}
[...] nos parece hoy que los números infinitamente pequeños e infinitamente grandes ... estándar. \par
Esto es obvio si introducimos ... \par
antes en este capítulo, \par
\begin{quoting}
no difiere del estilo de Arquímedes ...
arte de inventar
\end{quoting}
\end{quoting}
```

« [...] nos parece hoy que los números infinitamente pequeños e infinitamente grandes no son ni más ni menos reales que, por ejemplo, los números irracionales estándar.

»Esto es obvio si introducimos tales números axiomáticamente [...] Esta observación es igualmente cierta si nos aproximamos al problema desde el punto de vista del científico empírico [...] Pues, repitiendo una cita dada antes en este capítulo,

»“ no difiere del estilo de Arquímedes más que en las expresiones que son más directas en nuestro método y más conformes al arte de inventar ”.»

Comillas de seguir con spanish; la cita es una traducción y reelaboración, realizada por los autores, de la obra maestra *Non-standard analysis*, de Abraham Robinson, 1965 (Revised Edition, Princeton University Press, 1996)

¿Qué hace el entorno quoting? Simplemente incluye su cuerpo (*Texto*) entre comillas latinas, con la particularidad de que inserta «comillas de seguir» al inicio de cada párrafo del *Texto*, como en el ejemplo 1.5. Por sí solo no inicia un nuevo párrafo y puede ser anidado, para evitar problemas,

no más de tres veces. Las comillas de abrir y cerrar en los tres niveles se obtienen mediante los comandos

\lquoti	\rquoti
\lquotii	\rquotii
\lquotiii	\rquotiii

que corresponden a latinas, inglesas dobles e inglesas simples (de abrir, a izquierda, y cerrar); pueden ser redefinidos en cualquier momento.

Desactivación

Dado que spanish adopta varias medidas generales sobre el formato del documento e introduce numerosos atajos de teclado, que pueden resultar complicados para el usuario y tener incompatibilidades con otros paquetes, es bueno conocer que spanish proporciona vías para desactivar cada una de sus acciones. He aquí, reunidas, estas vías.

Empecemos por la más general. Para evitar que spanish tome las decisiones que toma al ser idioma principal, disponemos de dos formas: o bien redefinimos el grupo \layoutspanish, desde luego en el preámbulo, mediante

```
\renewcommand*{\layoutspanish}{}{}
```

o bien se inserta en el preámbulo el comando

```
\selectspanish*
```

Algo análogo puede hacerse para cada uno de los tres grupos restantes de comandos. Por ejemplo, si queremos eliminar los extras, o si se desea volver a la definición estándar de LATEX para \%, basta redefinir el macrocomando \textspanish, en una forma análoga:

```
\renewcommand*{\textspanish}{}{}
```

Refiriendo esta desactivación a \shorthandsspanish, observe que ¡por fin! hemos dado una explicación a la extraña quinta línea del código de nuestro preámbulo estándar, introducido allá por la página 18: eliminaba los métodos taquigráficos del español, algo muy sofisticado para un principiante en LATEX.

Si en lugar de desactivar todos los métodos taquigráficos tan sólo deseamos desactivar alguno de ellos, disponemos del comando

```
\spanishdeactivate{Car1 Car2 ...}
```

que desactiva todos los métodos taquigráficos que comiencen con los caracteres incluidos en su argumento, es decir, *Car1*, *Car2*, etc. Los métodos taquigráficos se activan al seleccionar el idioma; por tanto, si viniendo de otro idioma, seleccionamos el español, dichos métodos se reactivarán, y aunque alguno de ellos se haya desactivado con \spanishdeactivate, volverá a estar activo. Para que esto no ocurra es necesario «añadir» la desactivación al macrocomando:

```
\addto\shorthandsspanish{\spanishdeactivate{Car1 ...}}
```

La ventaja de este comando frente al genérico de babel \shorthandoff es que permite desactivar sólo los métodos de spanish y no los del idioma activo en el momento.

Disponemos de los siguientes «interruptores» específicos:

\deactivatetilden	\activatequoting	\deactivatequoting
-------------------	------------------	--------------------

El primero de ellos desactiva los métodos `~n` y `~N` que pueden presentar serios problemas de incompatibilidad con otros paquetes. Los otros dos activan o desactivan, respectivamente, los métodos `<>` y `>>` que tienen incompatibilidades casi seguras con los paquetes `ifthen`, `amstex` y con documentos que contengan código HTML o PDF. El entorno `quoting` siempre estará disponible como tal, aunque se haya usado `\deactivatequoting`.

1.4. Gestión de tipos mediante paquetes

En la lección 5 hemos descrito los tres atributos principales de los tipos de letra (familia, perfil y grosor) indicando, en el apartado I.5.1, los comandos que permiten seleccionar tales atributos. También allí señalábamos que Donald E. Knuth creó los tipos que, por defecto, `TEX` utiliza y los llamó «Computer Modern Fonts». Los de texto están agrupados en tres familias: la roman con adornos (`rmfamily`), la sanserif sin adornos (`sffamily`) —ambas de espaciado proporcional— y una familia de caracteres monoespaciados (`ttfamily`) similar a las antiguas máquinas de escribir. Cada una de estas familias dispone de caracteres con perfiles rectos e inclinados, de grosor normal y negritas. Además están los tipos necesarios para las fórmulas matemáticas. En esta sección vamos a profundizar un poco más en el tema describiendo, en particular, la forma en que pueden ser utilizadas otras familias de tipos (para las que también usaremos el nombre de fuentes) además de las construidas por Knuth.

Si bien es cierto que las posibilidades de `LATEX`, en cuanto a utilización de tipografías diferentes, son muy amplias, no lo es menos que explorarlas en su totalidad requiere un nivel de conocimientos que sobrepasa el objetivo que nos hemos marcado al escribir este libro. Adoptaremos, pues, una fórmula de compromiso consistente en señalar sucintamente los elementos esenciales que `LATEX` utiliza para la gestión de los tipos y comentar algunos paquetes que permitan al usuario utilizar, de forma sencilla, diferentes familias de tipos.

1.4.1. Creación de tipos: Computer Modern Fonts y METAFONT

Cuando Knuth creó las fuentes «Computer Modern» no se limitó a dibujar dichos caracteres individualmente, sino que creó el programa `METAFONT`, que permite a los computadores generar los tipos a partir de instrucciones expresadas en un lenguaje especial. Del mismo modo que `TEX` es un programa cuyo propósito es componer documentos bonitos, `METAFONT` (que está incluido en las distribuciones de `TEX`) es un programa cuyo propósito es crear tipos para ser utilizados en los documentos que `TEX` compone. De este modo cualquier usuario que conozca el lenguaje `METAFONT` puede crear nuevos tipos para ser usados con `TEX` y, de hecho, así ha ocurrido: Leslie Lamport creó las fuentes para construir los entornos `picture` y algunos caracteres matemáticos; la American Mathematical Society implementó nuevos símbolos matemáticos; Hermann Zapf diseñó las fuentes matemáticas Euler; el propio Knuth construyó una variante de las fuentes Computer Modern conocidas con el nombre de Concrete; Silvio Levy [40] creó fuentes para escribir en griego

clásico y moderno, etc.; pero también se han construido fuentes con caracteres ornamentales, cirílicos, arábigos, musicales, códigos de barras, fonéticos, ajedrecísticos o astronómicos.

Para que L^AT_EX pueda utilizar los tipos así creados es necesario indicarle la forma de hacerlo, por lo que, junto al diseño de los caracteres, se construye, habitualmente, un paquete que permite una utilización sencilla de dichos tipos. El resultado final es, pues, un trabajo conjunto de T_EX y METAFONT que cualquier programa de impresión o visualización desarrollado para T_EX es capaz de realizar.

La construcción de una fuente con METAFONT puede ser esquematizada diciendo que sobre un fichero de entrada, que contiene la descripción geométrica de cada uno de los «caracteres», actúa el compilador METAFONT para producir un fichero TFM con las dimensiones individuales de los caracteres y un fichero PK con el mapa de puntos de los mismos, de acuerdo con la resolución y el tamaño que se le han pasado como parámetros.

El fichero de entrada MF es un fichero «sólo texto» con comandos u órdenes escritos en el lenguaje METAFONT. La descripción de este lenguaje y su sintaxis excede el alcance de este libro y el lector interesado puede consultar el libro de Knuth [33].

El fichero TFM («T_EX font metric») es el único que utilizará el compilador T_EX para componer un documento. Este fichero contiene las cuatro dimensiones de cada carácter que T_EX necesita: altura, profundidad, anchura y «corrección itálica» \v (véase la lección 5). Estas longitudes están expresadas en «unidades relativas», en el sentido de que no dependen del tamaño final que se vaya a utilizar para imprimir o visualizar los tipos. De ese modo, aunque para cada tamaño y cada resolución se requiere un fichero PK diferente, basta con un único fichero TFM para cada fuente.

En principio, las dimensiones de la caja de cada carácter y la corrección itálica es todo lo que necesita saber T_EX. Sin embargo, el fichero TFM puede contener otro tipo de información relacionada con la excelencia estética de T_EX: son los «kerning» y las «ligaduras». Los primeros corresponden a espacios (positivos o negativos) que T_EX introduce cuando determinados caracteres aparecen juntos. Las segundas corresponden a la sustitución de determinadas combinaciones de caracteres por un nuevo símbolo; el ejemplo típico es la sustitución del binomio fi (f seguido de i) por el símbolo fi (véase la figura 1.2 en la página 188), o la sustitución de dos guiones -- por un solo guion – de mayor longitud. La información contenida en los ficheros TFM está en un formato binario para acelerar el acceso de T_EX a la misma, pero es posible obtener, a partir de ellos, mediante el programa TFTOPL, una versión en formato PL que es un formato «sólo texto» escrito en un cierto lenguaje. Recíprocamente, la conversión desde formato PL a formato TFM se realiza con el programa PLTOTF.

El fichero PK es un fichero de mapa de puntos que utilizan los programas encargados de dibujar, en la impresora o en el monitor, los caracteres de la fuente que aparece en el fichero DVI, entintando o no cada uno de los puntos contenidos en la caja correspondiente a un carácter que T_EX ha posicionado en un lugar concreto tras la composición. Se trata también de un fichero en formato binario para acelerar el acceso al mismo del programa de impresión o visualización. Hablando en sentido estricto, el compilador METAFONT no genera por sí mismo el fichero PK, sino que genera un fichero intermedio GF («generic font») que el programa GFTOPK convierte al formato PK, que es el formato que pueden utilizar todos los programas modernos para imprimir o visualizar ficheros DVI. El fichero GF puede ser utilizado por otros programas con propósitos

diferentes. Los programas actuales de impresión o visualización (DVIPS, YAP, XDVI...) suelen disponer de los recursos necesarios para realizar por sí mismos llamadas al compilador METAFONT y a GFTOPK para construir los ficheros PK que no han sido generados con anterioridad, colocándolos en el lugar adecuado. De ese modo, en general, el usuario no tendrá que preocuparse de estas cuestiones relativas a las fuentes y podrá concentrarse desde el primer momento en los aspectos de la composición. Las distribuciones L^AT_EX incluyen e instalan los siguientes elementos para las fuentes Computer Modern:

1. los ficheros TFM que L^AT_EX necesita para poder compilar el documento;
2. los ficheros MF que METAFONT necesita para poder generar las fuentes PK necesarias para ver e imprimir el resultado de la compilación;
3. los ficheros FD que contienen la información necesaria para seleccionar la fuente correcta cuando el usuario emplea comandos como \itshape, \bfseries, \textsc, etc., cuyo efecto fue explicado en la lección 5.

1.4.2. Fuentes EC y el paquete fontenc

En el apartado I.4.4 indicábamos que L^AT_EX es capaz de aceptar caracteres acentuados en el fichero fuente (cargando el paquete inputenc) y producir la salida esperada en los idiomas europeos. Sin embargo, las fuentes Computer Modern no disponen de caracteres acentuados, aunque sí incluyen de forma independiente los símbolos y tildes requeridos para su construcción. Es L^AT_EX el que se ocupa de generar el carácter acentuado mediante un proceso que podríamos esquematizar considerando dos etapas, en la primera de las cuales sitúa el carácter sin tilde, después retrocede lo que convenga e inserta la tilde, para a continuación alcanzar la posición que tenía antes de iniciar el retroceso.

En la conferencia de usuarios de T_EX en Cork (1990), se aprobó una nueva codificación para las fuentes de T_EX, más extensa que la inicial de 128 caracteres desarrollada por Knuth, que incorporaba los caracteres con sus tildes y cubría así más de treinta lenguas diferentes, incluyendo un total de 256 caracteres. Fue necesario extender las fuentes originales Computer Modern para texto a fin de que incluyeran directamente los caracteres con los diferentes formatos de tilde requeridos en las distintas lenguas. Esa nueva familia de tipos constituye lo que se llama las fuentes EC y, aunque todavía muchos usuarios de L^AT_EX siguen usando la antigua codificación, que no incorpora caracteres acentuados, la nueva tecnología tiene ventajas sobre la anterior, una de las cuales es un mejor comportamiento en relación con la segmentación silábica de palabras que incorporan tildes. Las distribuciones actuales de L^AT_EX permiten utilizar indistintamente el viejo modelo de codificación, llamado «OT1», y el nuevo, que se conoce con el nombre de «T1». Existen otras codificaciones «OT2», «T3»..., correspondientes a lenguas de caracteres no latinos, como cirílico, árabe, chino, etc. Por razones históricas de compatibilidad, L^AT_EX utiliza, salvo que se declare explícitamente otra cosa, la codificación «OT1». A esta codificación «OT1» es a la que se aplica la descripción que hemos realizado anteriormente sobre el proceso que L^AT_EX utiliza para generar un carácter acentuado; con «T1» se limita a seleccionar el carácter acentuado de la fuente correspondiente.

El lector no debe confundir la cuestión de la codificación, relacionada con la tecnología interna empleada por L^AT_EX para seleccionar familias de tipos y sobre la cual puede actuarse a través del paquete fontenc, con la posibilidad de poderse comunicar con L^AT_EX empleando caracteres acentuados en el texto fuente, sobre la cual se actúa a través del paquete inputenc, que informa a L^AT_EX de los códigos empleados por el teclado de nuestro computador. Ambas cuestiones son completamente independientes (véanse los apartados I.2.5 y I.4.4).

Las fuentes EC fueron creadas con METAFONT y actualmente vienen incorporadas en las distribuciones de L^AT_EX. Para usarlas con los idiomas europeos occidentales basta incorporar el paquete fontenc con la opción T1, es decir, \usepackage[T1]{fontenc}.

Los elementos que constituyen una fuente de tipos pueden ser visualizados en una tabla como las que aparecen en la figura 1.2 de la página 188, que recoge las tablas correspondientes a las fuentes Computer Modern Roman para las codificaciones «OT1» y «T1». Cada uno de los caracteres está identificado mediante un número en dicha tabla. La forma en que se realiza dicha identificación será descrita en el apartado 1.4.5; por ahora será suficiente que el lector se percate, comparando ambas tablas, de que la primera tiene 128 (= 2⁷) celdillas mientras que la segunda tiene el doble, 256 (= 2⁸); en ambos casos, sin contar los cuatro bordes exteriores que tienen que ver con la numeración, a la que antes aludíamos, y que la segunda incorpora los caracteres con tilde. Los números 7 y 8 que acaban de aparecer justifican el que, en ocasiones, a las fuentes OT1 se las denomine fuentes de 7 bits y a las «T1» fuentes de 8 bits.

1.4.3. Tipos PostScript

Cuando L^AT_EX acaba su trabajo de compilación ha compuesto el documento, lo cual significa que ha posicionado los caracteres (y los demás objetos que eventualmente pudiera haber en el documento) cubriendo líneas y páginas de una forma eficaz y elegante: el resultado de ese trabajo es un fichero DVI o PDF. Pero si reflexionamos un instante nos damos cuenta de que para posicionar un gráfico, una letra, etc., realmente lo único que importa son las dimensiones del gráfico o la letra; los contenidos de tales objetos son irrelevantes durante el trabajo de composición de líneas y páginas. El contenido concreto de cada una de esas cajas sólo adquiere relevancia en el momento de ver o imprimir el resultado de la composición, es decir, es algo relacionado con los programas que se ocupan de esas tareas y no con L^AT_EX. Tales programas habrán de ser capaces de visualizar o imprimir el gráfico y cada uno de los caracteres en la posición que tienen asignada en el fichero DVI o PDF. Por lo que al gráfico se refiere, necesitará ser de un formato que el programa de visualizar o imprimir sepa entender (ese tema ya lo hemos tratado en la lección 9), y en cuanto a los caracteres se refiere, ha llegado el momento de ver el contenido preciso de cada una de las cajas; ya no bastan sus dimensiones: el programa necesita llamar a METAFONT para que construya las ficheros PK necesarios para poder ver o imprimir los tipos correspondientes, y además ha de indicarle la resolución que necesita la fuente, pues el número de puntos por unidad de superficie disponibles en una pantalla de computador es mucho menor que el número de puntos disponibles en una impresora láser o en los fotolitos de una imprenta, y en cada caso conviene obtener la mejor calidad alcanzable.

Possiblemente se esté preguntando: si hasta ahora yo he podido ver e imprimir sin dificultad mis documentos, ¿a qué viene todo esto? ¡Ya sabrán mis programas lo que tienen que pedir a METAFONT! ¡Que se lo pidan con la resolución que necesiten y me dejen en paz a mí! Tiene toda la razón. De hecho, si le proporciona a otra persona que tenga un sistema L^AT_EX su texto fuente o su fichero DVI y esa persona tiene una impresora con una resolución de varios miles de puntos, su programa de impresión se ocupará de pedirle a METAFONT que le genere las fuentes PK con la resolución necesaria para tal impresora. Pero esos ficheros de nada le sirven a alguien que no tenga un sistema T_EX. En cambio, si le proporciona un fichero PDF, sin necesidad de tener instalado un sistema T_EX, podrá leerlo utilizando para ello ACROBAT READER o alguna herramienta similar. Pero para eso es necesario que el fichero PDF lleve incrustados los tipos Computer Modern. Estos tipos han sido generados por METAFONT en un sistema T_EX concreto a una resolución determinada y, por tanto, aunque se impriman en una impresora de miles de puntos, si fueron generados para una impresora con una resolución modesta, la calidad final será la misma en ambas impresoras. Además, el nivel de calidad con el que se visualizan en la pantalla del computador los ficheros PDF que utilizan fuentes PK es bajo.

Una vez planteado el problema, es claro que su solución exige incorporar al fichero PDF, en lugar de las fuentes a una determinada resolución, el algoritmo que permita a ACROBAT READER generarlas a la resolución que sea necesaria, lo cual requiere que tal algoritmo ha de ser inteligible para ACROBAT READER. Las fuentes que cumplen esa función se llaman fuentes vectoriales o, más concretamente, fuentes PostScript «Tipo 1», por contraposición a las fuentes de mapa de puntos, como las PK, que se denominan de «Tipo 3».

Por la calidad visual con que se muestran los tipos cuando se amplía el texto mediante la herramienta «zoom», es posible determinar si un documento PDF utiliza fuentes de «Tipo 1», de «Tipo 3» o una mezcla de ambas. Pero la información precisa está disponible en el apartado «Archivo|Datos del documento|Fuentes» de los menús desplegables de ACROBAT READER. Las mismas ideas que acabamos de describir para el formato PDF son aplicables a los ficheros en formato PS y a los programas capaces de utilizar dicho formato. Resumiendo, cuando se deseé una salida en formato PDF o PS es recomendable utilizar las fuentes «Tipo 1» en sustitución de las fuentes «Tipo 3».

Los tipos Computer Modern «OT1» diseñados por Knuth para METAFONT fueron implementados por Blue Sky Research en formato PostScript y completados por Y&Y con las fuentes de la AMS y las Euler, que son fuentes complementarias para L^AT_EX (que no forman parte del diseño realizado por Knuth). Actualmente, esas fuentes en formato «Tipo 1», que en tiempo fueron comerciales, son de dominio público.

En el congreso europeo de T_EX celebrado en el año 2001, fue presentado T_EXTRACE, un programa de Szabó Péter que permite la construcción de fuentes «Tipo 1» utilizando la información contenida en los ficheros MF. Esto abre nuevas e importantes posibilidades para la generación de «buenos» ficheros PDF y PS. En particular, las fuentes EC han sido implementadas en formato «Tipo 1» bajo el nombre «cm-super», lo que, en nuestra opinión, contribuirá a incrementar el uso de las fuentes EC.

Para que los programas DVIPS, PDFL^AT_EXy DVIPDFM incorporen, a los respectivos ficheros PS y PDF que generen, las fuentes en formato «Tipo 1» es necesario:

- disponer de los ficheros en formato PFB de dichas fuentes;
- indicarles explícitamente a dichos programas que utilicen este formato, lo cual se consigue a través del fichero de configuración `psffonts.map` en el caso de DVIPS y PDFLATEX, y en el fichero `config` para DVIPDFM.

Actualmente las distribuciones de los sistemas T_EX suelen estar configuradas adecuadamente para utilizar fuentes «Tipo 1» con DVIPS y PDFLATEX; sin embargo, por la novedad, es posible que algún sistema T_EX no esté bien configurado para usar las fuentes «cm-super», aunque seguro que esto cambiará en un futuro no muy lejano.

1.4.4. Las 35 fuentes PostScript de Adobe

Casi todas las impresoras PostScript tienen, residentes en su memoria ROM, una colección de fuentes en formato «Tipo 1» que incluye:

- cuatro familias, de nombres Times Roman, Palatino, Bookman y New Century Schoolbook, cuyos tipos utilizan adornos y disponen de perfiles rectos, itálicos e inclinados, de series normal y negrita, y de versalitas;
- dos familias, de nombres Helvetica y Avant-Garde, cuyos tipos no llevan adornos y disponen de perfiles rectos, itálicos e inclinados, de series normal y negrita y de versalitas;
- una familia monoespaciada, de nombre Courier, de naturaleza similar a las viejas máquinas de escribir, cuyos tipos disponen de perfiles rectos, itálicos e inclinados, de series normal y negrita, y de versalitas;
- una fuente cursiva, de nombre Zapf Chancery, que no dispone de versalitas, ni admite variaciones en perfil o series;
- dos fuentes de símbolos, de nombres Symbol y ZapfDingbats, que incluyen letras griegas mayúsculas y minúsculas, algunos símbolos matemáticos y otros símbolos adicionales.

En la figura 1.1, pág. 183, se encuentra una muestra de los tipos para las familias correspondientes a los cuatro primeros apartados del listado anterior, y en la figura 1.3, pág. 190, aparecen los símbolos disponibles para las dos fuentes incluidas en el último apartado. En conjunto, considerando itálicas, negritas, etc., suman 35 fuentes, lo que justifica el título de esta sección, que será el nombre utilizado en lo sucesivo para referirnos genéricamente a dichas fuentes.

La cuestión natural es: ¿puede L_AT_EX utilizar esas fuentes? La respuesta es afirmativa: L_AT_EX puede utilizar estas 35 fuentes y cualesquier otras en formato «Tipo 1» de las múltiples que se pueden adquirir comercialmente. Pero, aunque no entremos en los detalles, conviene hacer algunas precisiones.

1. En primer lugar, la utilización de tipos no estándar puede llevar aparejado que tanto la compilación de los ficheros T_EX como los ficheros DVI obtenidos dependan del sistema T_EX utilizado y, consiguientemente, no puedan ser compartidos con otros usuarios.
2. Para que la compilación pueda ser realizada, L_AT_EX necesita disponer de los ficheros TFM para dichos tipos.

Fuentes Tipo 1 de Adobe para escribir texto

Times-Roman**ptm**

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Palatino**ppl**

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Bookman**pbk**

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

NewCenturySchoolbook**pnc**

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Helvetica**phv**

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

AvantGarde**pag**

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

*ZapChancery**pzc*

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Courier**pcr**

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Figura 1.1: Algunas de las 35 fuentes de Adobe. A la izquierda figura el nombre original de la fuente (Times-Roman, Palatino...) y a la derecha el nombre de la misma con el que L^AT_EX la identifica (ptm, ppl...). En la sección 1.4.6 se presentan los comandos para acceder directamente a una fuente específica

3. Hay que indicar a L^AT_EX que utilice dichas fuentes en lugar de las Computer Modern. Además puede ser necesario adaptar la página de códigos de la fuente a las páginas de códigos de L^AT_EX.
4. Una vez compilado nuestro documento, para poder generar los ficheros PS y PDF puede ser necesario configurar los ficheros de configuración usados por DVIPS (`config.ps`), PDFTEX (`pdftex.cfg`) y DVIPDFM (`config`).
5. Finalmente, es necesario disponer de los ficheros PFB correspondientes a dichas fuentes.

Para las fuentes comerciales cabe esperar que el fabricante haya resuelto adecuadamente todos los problemas que se relacionan en la lista anterior, con la salvedad obvia del primero de ellos. En lo que sigue nos limitaremos a las 35 fuentes de Adobe.

Para las 35 fuentes de Adobe las distribuciones actuales tienen resueltos todos los problemas que la lista anterior plantea, incluyendo el primero de ellos, a través de la utilización del esquema de selección de fuentes NFSS y PSNFSS desarrollado por Frank Mittelbach y Rainer Schöpf en 1989. Los usuarios pueden ahora utilizar de forma sencilla y compatible las 35 fuentes de Adobe con sólo cargar un paquete, a través del comando `\usepackage`, en el preámbulo del documento fuente. Con esa información los programas DVIPS y PDFLATEX se ocupan de incrustar en los correspondientes ficheros PS y PDF las fuentes que no pertenezcan a las 35 fuentes de Adobe, junto con las instrucciones de uso para estas últimas.

Paquetes para usar las 35 fuentes de Adobe

La instalación de PSNFSS implementa varios paquetes; el cuadro 1.3 recoge los más interesantes, que permiten al usuario utilizar las 35 fuentes de forma sencilla. La primera fila indica el comportamiento por defecto de L^AT_EX con las fuentes Computer Modern. Cuando una celda está vacía significa que el paquete no modifica la familia de tipos que estuviesen activos (los Computer Modern, salvo que se haya cargado otro paquete). Así, el efecto de cargar un paquete es el de reemplazar una o varias de las familias Computer Modern por otras familias de Adobe. Ninguno de los paquetes listados cambian la codificación y, por defecto, usan «OT1»; sin embargo, es aconsejable utilizar la codificación «T1» mediante `\usepackage[T1]{fontenc}`.

Los dos últimos (times y palatino) están en desuso y se han añadido por razones históricas de compatibilidad. Ambos utilizan por defecto las fuentes matemáticas de la familia Computer Modern. El efecto de esta mezcla no es completamente satisfactorio (ni siquiera usando el paquete times, que es el que tiene unos caracteres sin adornos más parecidos a los de Computer Modern) debido a que el grosor de los caracteres, la altura media y el aspecto general no son homogéneos. Pero incluso, aun sin usar fórmulas matemáticas, la integración de la fuente Helvetica con Times Roman o Palatino es deficiente, debido a que la fuente Helvetica es algo mayor que las otras fuentes de Adobe para un mismo tamaño nominal. Por este motivo, el paquete helvet dispone de una opción para cargarlo con un cierto factor de escala. Dicha opción, scaled, introduce un factor de escala 0,95, que resulta aceptable para su armonización con los restantes tipos de Adobe. Dicho valor puede ser personalizado al cargar el paquete, pues la opción permite incluir un parámetro. El valor 0,92 resulta mejor adaptado al tamaño de la familia Times Roman y se declara mediante

```
\usepackage[scaled0.92]{helvet}
```

Con los paquetes `mathptmx` y `mathpazo` se consigue mejorar la armonía entre los tipos para matemáticas y los tipos para texto. El primero de ellos sustituye, a través de fuentes virtuales, la mayor parte de los caracteres matemáticos de las fuentes Computer Modern por caracteres de las fuentes Times Roman Italic y Symbol. La mayor limitación procede de la inexistencia de una fuente negrita para Symbol. En cambio, está accesible la opción `slantedGreek`, que produce letras griegas mayúsculas itálicas, aunque impide utilizar las letras griegas mayúsculas ordinarias, lo cual no puede interpretarse más que como un olvido en el diseño del paquete. Para evitar este inconveniente tenemos que modificar ligeramente el paquete `mathptmx`,² de la manera que se indica a continuación.

1. Eliminamos la actual definición de la opción `slantedGreek`, es decir, suprimimos completamente el comando `\DeclareOption{slantedGreek}{...}`.
2. Incluimos una nueva definición de dicha opción como sigue:

```
\DeclareOption{slantedGreek}{%
\let\Gamma\varGamma
\let\Delta\varDelta
\let\Theta\varTheta
\let\Lambda\varLambda
\let\Xi\varXi
\let\Pi\varPi
\let\Si\varSigma
\let\Upsilon\varUpsilon
\let\Phi\varPhi
\let\Psi\varPsi
\let\Omega\varOmega}
```

3. Añadimos los comandos necesarios para acceder tanto a los perfiles rectos como a los itálicos de las fuentes griegas mayúsculas:

```
\DeclareMathSymbol{\varGamma}{\mathalpha}{letters}{0}
\DeclareMathSymbol{\varDelta}{\mathalpha}{letters}{1}
\DeclareMathSymbol{\varTheta}{\mathalpha}{letters}{2}
\DeclareMathSymbol{\varLambda}{\mathalpha}{letters}{3}
\DeclareMathSymbol{\varXi}{\mathalpha}{letters}{4}
\DeclareMathSymbol{\varPi}{\mathalpha}{letters}{5}
\DeclareMathSymbol{\varSigma}{\mathalpha}{letters}{6}
\DeclareMathSymbol{\varUpsilon}{\mathalpha}{letters}{7}
\DeclareMathSymbol{\varPhi}{\mathalpha}{letters}{8}
\DeclareMathSymbol{\varPsi}{\mathalpha}{letters}{9}
```

²Recuerde que para llevar a cabo estas modificaciones debe cambiar de nombre el fichero y utilizar el paquete con nombre modificado.

```
\DeclareMathSymbol{\varOmega}{\mathalpha}{letters}{10}
\let\upGamma\Gamma
\let\upDelta\Delta
\let\upTheta\Theta
\let\upLambda\Lambda
\let\upXi\xi
\let\upPi\pi
\let\upSigma\sigma
\let\upUpsilon\Upsilon
\let\upPhi\phi
\let\upPsi\psi
\let\upOmega\Omega
```

A continuación vendría en el paquete el comando \ProcessOptions.

De esta forma tendremos declarados tanto los símbolos rectos \upGamma, \upDelta... como los itálicos \varGamma, \varDelta... Por otra parte, los comandos ordinarios \Gamma, \Delta... producirán símbolos rectos salvo cuando la opción slantedGreek esté activada, en cuyo caso producirán símbolos itálicos.

El paquete *mathpazo* utiliza Palatino para el texto base y sustituye la mayor parte de los caracteres matemáticos de las fuentes Computer Modern, vía fuentes virtuales, por caracteres de las fuentes Palatino y cinco nuevas fuentes matemáticas Pazo, en formato PFB, desarrolladas para ese fin; se da la circunstancia de que tales fuentes incluyen un símbolo para el euro, al que se puede acceder mediante el comando \ppleuro.

Disponibilidad de las 35 fuentes en el computador

La lectura con ACROBAT READER de los ficheros PDF no entraña ninguna dificultad, pues este programa incorpora las 35 fuentes y, en caso necesario, PDFT_EX ha incrustado los ficheros PFB para las fuentes que no pertenezcan a ese grupo.

La situación es análoga con las impresoras PostScript, pues éstas incorporan en su memoria ROM la información correspondiente a las 35 fuentes y un intérprete del lenguaje PostScript, por lo que cuando DVIPS genera un fichero PS no necesita incorporar dichas fuentes; basta con que contenga, en dicho lenguaje, la información necesaria para poder utilizar, en el momento de imprimir, las que están residentes en la impresora. Pero tal fichero PS sería inutilizable en impresoras de otro tipo, y no podríamos verlo en la pantalla del computador a menos que dispusiéramos de las correspondientes fuentes PFB.

Afortunadamente esa contrariedad puede soslayarse usando GHOSTSCRIPT, un programa que, como la impresora, es capaz de interpretar el lenguaje PostScript, produciendo en la pantalla del computador una imagen equivalente a la que se generaría en la impresora PostScript; además, la salida puede dirigirse a multitud de impresoras aunque no sean PostScript. El programa ALADDIN GHOSTSCRIPT está implementado en todas las plataformas y es gratuito.

Para poder realizar esta «suplantación» de la impresora, GHOSTSCRIPT necesita incorporar, como aquélla, las 35 fuentes de Adobe o al menos una versión «clónica» de las mismas, opción a

Paquete	rmfamily	sffamily	ttfamily	Matemáticas
mathptmx	CM Roman	CM sanserif	CM Typewriter	≈ CM Roman
mathpazo	Times Roman			≈ Times Roman
helvet	Palatino	Helvetica		≈ Palatino
avant		AvantGarde		
chancery	Zapf Chancery			
bookman	Bookman	AvantGarde	Courier	
newcent	NewCenturySchoolbook	AvantGarde	Courier	
courier			Courier	
times	Times Roman	Helvetica	Courier	
palatino	Palatino	Helvetica	Courier	

Cuadro 1.3: Paquetes para las 35 fuentes PostScript

la que recurre debido a que las de Adobe no son fuentes gratuitas de libre distribución (o al menos no lo eran, ya que, ahora, ACROBAT READER, que sí es gratuito, las incorpora).

Estas fuentes clónicas que GHOSTSCRIPT utiliza fueron desarrolladas por URW++ Design and Development Incorporated de Hamburgo (<http://www.urwpp.de>) y se denominan, genéricamente, fuentes URW. Tienen nombre diferente de las de Adobe, pero una tabla informa a GHOSTSCRIPT de las sustituciones a realizar (por ejemplo, sustituye Times Roman por n0210031.pfb).

Conviene advertir que, aunque los ficheros en formato DVI generados utilizando las 35 fuentes de Adobe sean independientes del sistema \TeX en relación con DVIPS, no lo son en relación con los programas para ver ficheros en formato DVI. Más específicamente, algunos visores de archivos DVI (entre ellos YAP, XDVI y KDVI) que sólo pueden utilizar fuentes en formato PK, están preparados para generar de manera automática las fuentes en dicho formato a partir del formato PFB, en caso de no estar disponible la fuente en el sistema en formato MF. Pero hay otros visores para los que son inservibles (al menos inicialmente) los ficheros DVI que utilizan elementos de las 35 fuentes de Adobe. Esto no se debe a que el visor sea incapaz de generar las fuentes en formato PK, sino a que, aunque éstas estén presentes en el sistema, son incapaces de utilizarlas porque se emplean mecanismos de fuentes virtuales que no todos los programas de visionado soportan (los indicados al comienzo de este párrafo sí lo hacen).

1.4.5. Tabla de una fuente y acceso a los caracteres

Una fuente puede ser representada mediante una tabla o matriz en la que cada «carácter» está ubicado en una celda identificable mediante un número. En la figura 1.2 mostramos dos de esas tablas, una con la codificación «OT1» (cmr10) y otra con la codificación «T1» (ecrm10), para la fuente Computer Modern Roman, que es la que \LaTeX usa por defecto para el texto normal.

Estas tablas pueden construirse compilando con \LaTeX el archivo nfssfont.tex y respondiendo interactivamente a la pregunta que solicita el nombre de la fuente a procesar. Para crear la

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Τ	"0x
'01x	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl	
'02x	ι	ј	΄	΄	΄	΄	΄	΄	"1x
'03x	ȝ	ȝ	æ	œ	ø	Æ	Œ	Ø	
'04x	ȝ	!	”	#	\$	%	&	,	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	ı	=	ł	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[“]	^	·	
'14x	‘	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	—	—	”	~	..	
	"8	"9	"A	"B	"C	"D	"E	"F	
	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	΄	΄	^	‐	΅	΅	΅	΅	"0x
'01x	΅	΅	·	·	΅	,	΅	΅	
'02x	΅	΅	΅	΅	΅	—	—	—	"1x
'03x	ο	ι	ј	ff	fi	fl	ffi	ffl	
'04x	ȝ	!	”	#	\$	%	&	,	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	˂	=	˃	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[\]	^	—	
'14x	‘	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	{		}	~	-	
'20x	Ā	Ā	Ć	Ć	Đ	Ē	Ē	Ğ	"8x
'21x	Ł	Ł	Ł	Ń	Ń	Đ	Ó	Ŕ	
'22x	Ř	Ř	Š	Š	Ť	Ť	Ů	Ů	"9x
'23x	Ŷ	Ž	Ž	Ž	IJ	İ	đ	§	
'24x	ă	ă	ć	ć	đ	ě	ę	ğ	"Ax
'25x	í	í	ł	ń	ň	յ	ő	ŕ	
'26x	ř	š	š	š	ť	ķ	ú	ú	"Bx
'27x	ÿ	ž	ž	ž	ij	î	ż	ł	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ҫ	"Cx
'31x	È	É	Ê	Ë	Ï	Í	Î	Ï	
'32x	Ð	Ñ	Ò	Ó	Ӯ	Ӯ	Ӯ	Ӯ	"Dx
'33x	Ӯ	Ӯ	Ӯ	Ӯ	Ӯ	Ӯ	Ӯ	Ӯ	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	ð	ñ	ò	ó	ö	ö	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ß	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figura 1.2: Tablas de caracteres para Computer Modern Roman en las codificaciones «OT1» y «T1»

tabla se introduce `\table` después de la línea de comandos del compilador TeX mostrada como un asterisco *. Para acabar el proceso se introduce `\stop`. La compilación interactiva del archivo `nfssfont.tex` permite utilizar otros comandos (que pueden ponerse uno a continuación del otro) además de `\table` y `\stop`; para tener la lista completa basta introducir el comando `\help` después del asterisco.

Para incorporar un determinado carácter de la fuente en uso pueden utilizarse los comandos

<code>\symbol{Número}</code>	<code>\charNúmero</code>
------------------------------	--------------------------

siendo *Número* el valor que identifica a dicho carácter. Los caracteres se van numerando de izquierda a derecha y de arriba abajo en la forma usual. Hay tres maneras diferentes de realizar la asignación numérica, *Número*, a las celdas de dicha tabla: en formatos decimal (base 10), octal (base 8) o hexadecimal (base 16).

- En la numeración decimal se asignan a la primera celda los dígitos 000; así, por ejemplo, el último carácter de la primera fila es el 007 y el penúltimo de la segunda fila es el 014. Para insertar dichos caracteres para la fuente en uso bastaría con escribir `\symbol{007}` y `\symbol{014}`, respectivamente. O, si se prefiere, `\char007` y `\char014`, con una estructura sintáctica menos usual en L^AT_EX.
- En la numeración octal se asignan a la primera celda los dígitos 000 y se numera en la forma usual, sólo que utilizando únicamente los caracteres 0 a 7. Así, por ejemplo, el último carácter de la primera fila sigue siendo el 007, mientras que el primero de la segunda fila es el 010 y el penúltimo de la segunda fila es el 016. Si se utiliza esta forma de numerar, el *Número* debe estar precedido de un apóstrofo ('), lo que para el ejemplo antes considerado significa `\symbol{'007}` y `\symbol{'016}`.
- En la numeración hexadecimal se asignan a la primera celda los dígitos 000, numerando en la forma usual, sólo que ahora, en lugar de tener 10 caracteres para numerar (de 0 a 9), como ocurría en la numeración decimal, u ocho caracteres (de 0 a 7), como en la numeración octal, disponemos de dieciséis caracteres: los diez primeros son del 0 al 9 y los siguientes son A, B, C, D, E, F. En este caso son necesarios únicamente dos dígitos (en lugar de tres) para numerar las celdas de la tabla, y el argumento *Número* debe ir precedido del símbolo de las dobles comillas ("). Volviendo al ejemplo, la numeración hexadecimal de los caracteres considerados en el punto anterior es, respectivamente, "07 y "0E.

Ahora entenderá mejor la necesidad de indicar a L^AT_EX la página de códigos que corresponde a una determinada fuente. Y le resultará fácil comprender que cuando usted escribe `\beta`, lo que L^AT_EX hace (entre otras cosas) es seleccionar el carácter que corresponde a un cierto *Número* en la tabla de una determinada fuente que contiene las letras griegas usadas en matemáticas.

Dos fuentes de símbolos: los paquetes pifont y marvosym

Con los paquetes pifont y marvosym es fácil utilizar en nuestro documento las fuentes Symbol, ZapfDingbats (parte de las 35 fuentes de Adobe) y Marvosym, creada por Martin Vogel en formato True Type y luego transformada al formato «Tipo 1» por Thomas Henlich (véanse las figuras 1.3 en la página siguiente y 1.4 en la página 193).

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!	∀	#	Ξ	%	&	Ξ	'2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	'3x
'07x	8	9	:	;	<	=	>	?	
'10x	≡	A	B	X	Δ	E	Φ	Γ	'4x
'11x	H	I	ø	K	Λ	M	N	O	
'12x	Π	Θ	P	Σ	T	Y	ς	Ω	'5x
'13x	Ξ	Ψ	Z	[⋮]	⊥	-	
'14x	-	α	β	χ	δ	ε	φ	γ	'6x
'15x	η	ι	φ	κ	λ	μ	ν	ο	
'16x	π	θ	ρ	σ	τ	υ	ω	ω	'7x
'17x	ξ	ψ	ζ	{		}	~		
'24x	Υ	'	≤	/	∞	f	♣	'Ax	
'25x	♦	♥	♠	↔	←	↑	→	↓	
'26x	°	±	"	≥	×	∞	∂	•	'Bx
'27x	÷	≠	≡	≈	...		—	↓	
'30x	¤	⌚	⌚	⌚	⊗	⊕	∅	∩	'Cx
'31x	∪	▷	≡	◁	⊂	⊆	∈	∉	
'32x	∠	∇	®	©	™	Π	√	.	'Dx
'33x	¬	^	∨	↔	≤	↑↑	⇒	↓↓	
'34x	◊	⟨	®	©	™	Σ	()	'Ex
'35x	ƪ	⌜		⌞	⌞	⌞	⌞	⌞	
'36x		⟩	∫	∫		J)		'Fx
'37x)	˥		˩		﴿	˩		
	"8	"9	"A	"B	"C	"D	"E	"F	
	'0	'1	'2	'3	'4	'5	'6	'7	
'04x	✖	✖	✖	✖	✖	✖	✖	✖	'2x
'05x	✖	✖	✖	✖	✖	✖	✖	✖	
'06x	✖	✖	✖	✓	✓	✖	✖	✖	'3x
'07x	✖	✖	✖	✖	✖	✖	✖	✖	
'10x	✖	✖	✖	✖	✖	✖	❖	❖	'4x
'11x	★	★	★	★	★	★	★	★	
'12x	★	★	★	*	*	*	*	*	'5x
'13x	*	*	*	*	*	*	*	*	
'14x	*	*	*	*	*	*	*	*	'6x
'15x	*	*	*	*	●	○	■	□	
'16x	□	□	□	▲	▼	◆	❖	♦	'7x
'17x				‘	,	“	”		
'24x	¶	¶	¶	¶	♥	♦	◎	¤	'Ax
'25x	♣	♦	♥	♠	①	②	③	④	
'26x	⑤	⑥	⑦	⑧	⑨	⑩	①	②	'Bx
'27x	③	④	⑤	⑥	⑦	⑧	⑨	⑩	
'30x	①	②	③	④	⑤	⑥	⑦	⑧	'Cx
'31x	⑨	⑩	①	②	③	④	⑤	⑥	
'32x	⑦	⑧	⑨	⑩	→	→	↔	↑↓	'Dx
'33x	↗	→	↗	→	→	→	→	→	
'34x	➡	➡	➢	➢	➢	➢	➢	➢	'Ex
'35x	➡	➡	➡	➡	➡	➡	➡	➡	
'36x	➡	➡	⌚	➡	➡	➡	➡	➡	'Fx
'37x	➡	➡	→	→	➡	➡	➡	➡	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figura 1.3: Tablas de caracteres para Symbol (psyr) y ZapfDingbats (pzdr)

El paquete pifont Este paquete permite un acceso sencillo a los símbolos de las fuentes Symbol y ZapfDingbats mediante el comando básico

```
\Pisymbol{Fuente}{Número}
```

donde *Fuente* admite uno de los valores psy o pzd y el argumento *Número* se comporta como en el comando \symbol anteriormente descrito.

```
\Pifill{Fuente}{Número}
```

```
\Piline{Fuente}{Número}
```

Estos comandos utilizan una sintaxis similar a la de \Pisymbol para, mediante la repetición de un mismo símbolo, llenar un espacio extensible y delimitado, el primero de ellos, o bien una línea entera, el segundo. También están implementados sendos entornos para construir listas cuyas viñetas utilizan las fuentes Symbol o ZapfDingbats.

```
\begin{Pilist}{Fuente}{Número}
\item Primer ítem
...
\item N-ésimo ítem
\end{Pilist}
```

```
\begin{Piautolist}{Fuente}{Número}
\item Primer ítem
...
\item N-ésimo ítem
\end{Piautolist}
```

En el entorno Pilist el símbolo permanece fijo, como ocurre en los entornos itemize, mientras que el entorno Piautolist funciona de forma similar a un entorno enumerate: se fija el *Número* que establece la viñeta del primer ítem y en cada nuevo ítem se imprime la viñeta que ocupa la posición siguiente en la tabla de la fuente. Ambos entornos están construidos utilizando el entorno list descrito en 3.2, siendo, de hecho, particularizaciones de dicho entorno general y, en consecuencia, son aplicables a estos entornos las personalizaciones allí descritas.

Como lo más frecuente es usar estos comandos y entornos con la fuente ZapfDingbats, están implementadas versiones simplificadas que utilizan únicamente el segundo argumento, ya que asignan al primero el valor pzd. Estas versiones son:

```
\ding{Número}
\dingfill{Número}
```

```
\dingline{Número}
```

```
\begin{dinglist}{Número}
\item Ítem primero
...
\item Ítem n-ésimo
\end{dinglist}
```

```
\begin{dingautolist}{Número}
\item Ítem primero
...
\item Ítem n-ésimo
\end{dingautolist}
```

EJEMPLO 1.6

El paquete ... siguientes idiomas:
\begin{dingautolist}{'266}
\item español;
...
\end{dingautolist}

El paquete babel permite gestionar los siguientes idiomas:
❶ español;
❷ catalán;
❸ gallego.

Listas con el paquete pifont

El paquete marvosym Este paquete implementa un comando básico

```
\mvchr{Número}
```

con la misma sintaxis del comando `\char` (anteriormente explicado) junto con una colección de comandos que consisten en poner nombre, en inglés, a cada uno de los símbolos de la tabla de la fuente utilizando dicho comando básico. Únicamente destacaremos el símbolo oficial del euro que ocupa la posición 68 en numeración decimal (equivalentemente 104 en octal) y al que puede accederse desde el paquete con el comando `\EUR`. El paquete `marvosym` se convierte, así, en una alternativa al paquete `eurosym`, que fue descrito en la lección 4, con la ventaja frente a aquél de disponer de más símbolos con un único paquete. Si se optara por esta solución parecería razonable crear un nuevo comando `\euro` (más fácil de recordar que el utilizado por el paquete `marvosym`) del siguiente modo:

```
\newcommand*{\euro}{\EUR}
```

La misma estrategia puede emplearse para asociar comandos a aquellos símbolos que sean de uso frecuente para el usuario, sin necesidad de acudir a los nombres utilizados por Thomas Henlich [29].

EJEMPLO 1.7

```
\usepackage{marvosym}
...
{\Large\mvchr'124} 968 869 123 \qquad
{\Large\mvchr'110} 630 640 650 \qquad
{\Large\mvchr'153} latex@um.es
```

☎ 968 869 123 ☎ 630 640 650 ✉ latex@um.es

Símbolos con el paquete `marvosym`

1.4.6. Un tipo para cada ocasión

En los directorios de CTAN existen multitud de fuentes. Unas están destinadas a escribir texto, y no sólo con caracteres romanos de diferentes diseños, sino también con caracteres cirílicos, griegos, árabes, etc. Otras fuentes son de tipo simbólico, por lo que permiten escribir códigos de barras, partituras musicales, símbolos fonéticos, braille... La mayor parte de estas fuentes van acompañadas de un paquete STY, con documentación para facilitar su uso, y de los archivos necesarios para su instalación. El fichero `symbols-a4.pdf`, distribuido con el conjunto de documentos denominado `comprehensive`, muestra una buena colección de fuentes.

Pero también pueden encontrarse fuentes cuyo autor se ha limitado a construir los ficheros MF; en ese caso el usuario ha de ocuparse del resto, lo cual requiere un mayor dominio del sistema NFSS del que se pretende alcanzar en este libro. Para acabar, señalamos que también es posible utilizar con L^AT_EX las fuentes «True Type», que se distribuyen con el sistema MS-WINDOWS o con programas para dicho sistema (nos ha aparecido un ejemplo a propósito del paquete `marvosym`), pero esa utilización aún requiere mayores conocimientos, porque es necesario empezar desde una etapa anterior que exige construir previamente los ficheros MF o PFB. El lector interesado puede encontrar información más específica, pero todavía básica, en nuestro anterior libro [11] o bien en el libro [21] de los creadores del sistema NFSS.

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	□	□	□	□	□	□	□	□	"0x
'01x	□	□	□	□	□	□	□	□	"1x
'02x	□	□	□	□	□	□	□	□	"2x
'03x	□	□	□	□	□	□	□	□	"3x
'04x	●	—	Δ	Δ	Δ	Δ	⌚	⌚	"4x
'05x	()	×	+	,	-	.	/		"5x
'06x	0	1	2	3	4	5	6	7	"6x
'07x	8	9	→	⇒	≤	≥	≥	↔	"7x
'10x	@	Ø	☒	CE	€	ƒ	ℳ	—	"8x
'11x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"9x
'12x	→	↔	---	↔	⌚	⌚	⌚	⌚	"Ax
'13x	✖	✖	✖	✖	✖	✖	✖	✖	"Bx
'14x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Cx
'15x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Dx
'16x	→	↔	---	↔	⌚	⌚	⌚	⌚	"Ex
'17x	✖	✖	✖	✖	✖	✖	✖	✖	"Fx
'20x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Ax
'21x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Bx
'22x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Cx
'23x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Dx
'24x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Ex
'25x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	"Fx
'26x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'27x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'30x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'31x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'32x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'33x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'34x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'35x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'36x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
'37x	⌚	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figura 1.4: Tablas de caracteres para marvosym (fmvr8x)

Utilización puntual de tipos

La forma adecuada para manejar las diferentes familias, perfiles, grosores y tamaños fue descrita en la lección 5. En la sección 1.4.4 se ha indicado el modo de reemplazar las fuentes de las familias Computer Modern (roman, sanserif, typewriter) por otras familias mediante la utilización de paquetes específicos.

Excepcionalmente puede desearse sobrepasar ese marco, ya sea para distinguir un párrafo mediante una fuente especial, que no forma parte de las familias roman, sanserif o typewriter en uso; para hacer un rótulo de un tamaño mayor del que permiten los comandos descritos en la lección 5; para modificar localmente la codificación o por cualquier otro motivo singular.

Los comandos de bajo nivel que permiten realizar tareas de ese tipo son los siguientes:

```
\fontencoding{Codificación}
\fontfamily{Familia}
\fontseries{Serie}
\fontshape{Perfil}
\fontsize{Tamaño}{Interlínea} \selectfont
```

- Los valores más comunes para *Codificación* son: OT1, T1, OT2...
- La *Familia* debe estar adecuadamente identificada en L^AT_EX; típicamente, la instalación de un paquete de fuentes habrá incorporado un fichero FD para gestionar la fuente (por ejemplo, *T1ptm.fd* para la familia Times-Roman de Adobe con codificación «T1»).
- Los valores más comunes para *Serie* son: *m* (medium = medio o normal), *b* (bold = negrita), *bx* (bold extended = negrita extendida), *sb* (semi bold = seminegrita) y *c* (condensed = condensada). No todos han de estar disponibles; el fichero FD contiene la información.
- Los valores más comunes para *Perfil* son: *n* (normal o recto), *it* (italico), *s1* (slanted = oblicuo o inclinado), *sc* (small caps = versalita).
- El argumento *Tamaño* es una longitud rígida; si la unidad no aparece especificada se asume que es *pt*. La longitud *Interlínea* admite valores elásticos; si se utilizan dichos valores es necesario indicar explícitamente la unidad, pero si se utilizan valores rígidos es de aplicación lo dicho para *Tamaño*. Típicamente, *Interlínea* suele ser un 20 % mayor que *Tamaño*. Conviene advertir que algunos ficheros FD pueden haber fijado los valores admisibles para *Tamaño* (así ocurre para las fuentes Computer Modern *t1cmr.fd*, *OT1cmr.fd*...); en tal caso L^AT_EX sustituirá automáticamente el valor *Tamaño* por el más cercano a éste entre los valores admisibles.
- Configurados estos parámetros (o algunos de ellos), el comando *\selectfont* los hace actuar (con carácter local si está delimitado por un grupo).

EJEMPLO 1.8

Aunque no se haya cargado ... es posible destacar
{\fontfamily{pzc}\fontsize{10.2pt}{12pt}
\selectfont
un texto utilizando la fuente ZapfChancery}.
En este caso ha sido necesario ...

Aunque no se haya cargado el paquete chancery es posible destacar *un texto utilizando la fuente ZapfChancery*. En este caso ha sido necesario utilizar el comando *\fontsize* debido a que el tamaño original de la fuente no armonizaba bien con el tamaño del texto base.

1.5. Más sobre contadores y longitudes

EN la lección 17 hemos estudiado las nociones básicas sobre contadores y longitudes ya existentes en L^AT_EX; en particular hemos aprendido la forma de conocer y gestionar los valores de dichos registros y, en el caso de los contadores, hemos manejado diferentes formatos y representaciones para los mismos. En esta sección aprenderemos a crear nuevos contadores, ya sean autónomos o «subordinados» a otros contadores existentes, y nuevas longitudes, profundizando en el manejo de las longitudes elásticas. Se introducirán nuevas unidades de longitud destinadas a expresar las holguras en los valores elásticos de una longitud; a diferencia de las unidades de

longitud que aparecen en la lección 17, las nuevas unidades no tienen un valor concreto, expresaible, por ejemplo, en mm, sino que su valor se adapta a las circunstancias; son, pues, unidades muy versátiles, o, si se prefiere, muy elásticas.

1.5.1. Contadores

A los contadores definidos en L^AT_EX se pueden añadir otros nuevos. Las razones para crear nuevos contadores están relacionadas, generalmente, con la creación de estructuras propias del usuario; en este apartado veremos algunos ejemplos sencillos que ilustran esta afirmación. Antes de describir la sintaxis para crear un nuevo contador, añadiremos un nuevo comando a los ya conocidos \setcounter y \addtocounter, que se describieron en la lección 17, que está destinado a recuperar el valor numérico de un contador, independientemente de su formato o representación.

`\value{NombreContador}`

El comando \value frecuentemente forma parte del argumento de otro comando para la gestión de valores de contadores, tal y como se ilustra en el ejemplo 1.9.

`\newcounter{NuevoContador}[ContadorExistente]`

Este comando introduce un contador con nombre *NuevoContador* y le asigna cero como valor inicial. En caso de que dicho contador ya existiera se produciría un mensaje de error.

Si se especifica en el argumento opcional un contador que ya existe, cuyo nombre es *ContadorExistente*, entonces el contador *NuevoContador* irá subordinado al *ContadorExistente*, en el sentido de que un incremento en el valor del *ContadorExistente* provocará que el valor del contador *NuevoContador* se reinicie a cero. Hay que advertir que los contadores que están subordinados a otro no se ponen a cero cuando se modifica el valor de éste con los comandos \addtocounter o \setcounter, descritos en la lección 17. Este objetivo se consigue con el comando \stepcounter, que se detalla más adelante.

Al introducir un nuevo contador se crea el comando \the*NuevoContador* correspondiente, cuya definición por defecto es \arabic{NuevoContador}. Así, aunque el *NuevoContador* esté subordinado a un *ContadorExistente*, en la representación del *NuevoContador* no aparece ninguna referencia al *ContadorExistente*, salvo que se modifique de forma explícita dicha definición.

EJEMPLO 1.9

```
\newcounter{conserva}
\begin{enumerate}
\item El cuerpo de los reales.
\item El cuerpo de los complejos.
\setcounter{conserva}{\value{enumi}}
\end{enumerate}
En las siguientes se estudian propiedades ...
\begin{enumerate}
\setcounter{enumi}{\value{conserva}}
\item Continuidad ...
\end{enumerate}
```

Enlazar varias listas de ítems manteniendo la numeración

Las primeras lecciones están dedicadas a los campos numéricos:

1. El cuerpo de los reales.
2. El cuerpo de los complejos.

En las siguientes se estudian propiedades básicas de las funciones:

3. Continuidad
4. Diferenciación

Los dos comandos que siguen son también muy útiles en la implementación de estructuras propias del usuario, a través de comandos y entornos que facilitan la realización de construcciones personales de uso habitual.

<code>\stepcounter{NombreContador}</code>	<code>\refstepcounter{NombreContador}</code>
<code>\stepcounter</code> Incrementa en una unidad el contador <i>NombreContador</i> y reinicia todos los contadores subordinados a él.	
<code>\refstepcounter</code> Es lo mismo que el anterior, pero también declara como valor del comando <code>\ref</code> el texto generado por <code>\theNombreContador</code> cuando se utilizan referencias cruzadas con los comandos <code>\label</code> y <code>\ref</code> (véase el apartado I.8.1).	

EJEMPLO 1.10

```
\newcounter{prg}[section]\newcounter{linea}[prg]
\newcommand*\lin{\%
    \addtocounter{linea}{1}\thelinea\quad}
\renewcommand*\theprg{\%
    \arabic{section}.\arabic{prg}}
\newenvironment*{progra}{%
\refstepcounter{prg}
\begin{center}Programa~\theprg\end{center}
\ttfamily\obeylines\obeyspaces\par}
\section{Programas desarrollados}
\begin{progra}
\lin una línea
\lin otra línea
\end{progra}
\begin{progra}
\lin una línea
\lin otra línea
\end{progra}
\begin{progra}\label{ML}
\lin más líneas
\lin y aún más
\end{progra}
\section{Programas en desarrollo}
\begin{progra}
\lin aún necesitamos más \lin y ya la última
\end{progra}\medskip ... al programa~\ref{ML}
```

1. Programas desarrollados

Programa 1.1

1 una línea
2 otra línea

Programa 1.2

1 más líneas
2 y aún más

2. Programas en desarrollo

Programa 1.3

1 aún necesitamos más
2 y ya la última

Observe cómo establecemos una referencia cruzada al programa 1.2

Creación de nuevos contadores y utilización del comando `\refstepcounter`. Sobre los comandos `\obeylines` y `\obeyspaces` véase el apartado PARA SABER MÁS al final del capítulo

1.5.2. Longitudes

Al igual que ocurre con los contadores, es posible añadir nuevas longitudes a las que ya están disponibles en L^AT_EX usando el comando

<code>\newlength{\NuevaLongitud}</code>

que crea una nueva longitud y se asigna a *NuevaLongitud*. Si tal comando ya existiera (aunque no fuese una longitud) L^AT_EX emitiría un mensaje de error. Vimos en el apartado I.17.2 que existen dos tipos de longitudes: las que admitían únicamente valores rígidos (llamadas en la terminología de T_EX *dimen*) y las que admitían también valores elásticos (llamadas *skip*). Las longitudes creadas con el comando `\newlength` pertenecen a estas últimas y al crearlas se les asigna un valor inicial de 0 pt.

Junto con los comandos `\setlength` y `\addtolength` introducidos en el apartado I.17.2 resultan también de utilidad para la gestión de longitudes los siguientes:

```
\settowidth{\NombreLongitud}{Objeto}
\settoheight{\NombreLongitud}{Objeto}
\settodepth{\NombreLongitud}{Objeto}
```

En este caso las longitudes no se asignan de forma directa, sino que se calculan como el valor de la anchura «width», altura «height» o profundidad «depth» de la caja que contiene un cierto *Objeto*.

EJEMPLO 1.11

```
\newlength{\MiLongi}\newlength{\OtraLongi}
En este ejemplo ... valor inicial:
valor de MiLongi = \the\MiLongi;
valor OtraLongi = \the\OtraLongi.
\settowidth{\MiLongi}{\itshape carretera}\par
Veamos ahora ... palabra {\itshape carretera}...
caja que resulta ser de \the\MiLongi.
\OtraLongi=2,5\MiLongi\par
Multiplicando por 2,5 el valor actual de
MiLongi y asignándolo a OtraLongi
se obtiene \the\OtraLongi. \par
Ahora, dentro del mismo párrafo...
\begin{minipage}{7mm}
  dos\\ líneas\end{minipage}
\settoheight{\MiLongi}{%
\begin{minipage}{7mm}
  dos\\ líneas\end{minipage}\end{minipage}}
...que resulta ser de \the\MiLongi ...
```

Nuevas longitudes y medición de objetos

En este ejemplo creamos dos nuevas longitudes y comprobamos su valor inicial: valor de MiLongi = 0.0pt; valor OtraLongi = 0.0pt.

Vemos ahora el modo en que se le asigna a una longitud la anchura de la caja que contiene un determinado objeto, que en nuestro caso será la palabra *carretera*, lo cual, en particular, nos permite medir la anchura de dicha caja que resulta ser de 33.5245pt.

Multiplicando por 2,5 el valor actual de MiLongi y asignándolo a OtraLongi se obtiene 83.81126pt.

Ahora, dentro del mismo párrafo, insertamos un entorno `minipage` que tiene dos líneas y medimos su altura que resulta ser de 10.84592pt, mientras que su profundidad es de 6.34592pt.

Longitudes muy elásticas

En el apartado I.17.2 introdujimos el concepto de valor elástico para longitudes (`skip`) capaces de admitir tales valores. Recordemos que a este tipo de longitudes se les asigna un valor de referencia y hasta un máximo de dos valores que determinan la capacidad de estiramiento y encogimiento de la longitud, en relación al valor de referencia; de ese modo se permite a L^AT_EX una cierta tolerancia en el valor que finalmente le asigne a la longitud en cada situación, que dependerá de las necesidades con las que se encuentre al tratar de distribuir el espacio de la mejor manera posible.

Aunque la noción de valor elástico de una longitud, con holgura controlada, resulta muy útil, no resuelve todas las necesidades. Hay ocasiones en las que se necesita introducir un espacio cuyo valor preciso se desconoce y para las que las unidades de longitud que aparecen en la tabla I.17.1 resultan insuficientes para proporcionar la máxima elasticidad requerida. Como ejemplo de tales situaciones podría servir el que empleábamos al comienzo del apartado I.18.5 y que consistía en resolver el problema de conseguir la máxima separación posible entre dos cajas situadas en una misma línea. Ese problema fue resuelto allí mediante el comando `\hfill`. Pero si tratamos de imaginar cómo podría ser definido un comando con esa función, es posible que llegáramos

(tecnismos aparte) a la solución adoptada por T_EX: insertar un espacio elástico, cuyo valor de referencia es 0 pt con una holgura lo suficientemente grande para ser útil en todas las situaciones y utilizar siempre el estiramiento máximo que permitan las condiciones disponibles. Para este tipo de elasticidades tan acomodables es necesario introducir nuevas unidades de longitud que se utilizan únicamente en la parte correspondiente a la holgura.

Las unidades

fil	fill
-----	------

cumplen este propósito. Se trata de unidades para la holgura que, sin tener un valor preciso, convierten las unidades de la tabla I.17.1 en despreciables frente a ellas, pues mientras que las que aparecen allí tienen un valor concreto, las nuevas unidades crecen y crecen, según el contexto. Las dos unidades anteriores han sido creadas para tener dos grados de «elasticidad infinita»: la primera es una unidad de longitud elástica infinitamente más grande que cualquier longitud rígida dada, mientras que la segunda es una unidad de longitud elástica infinitamente más grande que la primera (y en consecuencia que las unidades rígidas).

Existen varios comandos destinados a simplificar la utilización de las unidades fil y fill que iremos describiendo en el resto de esta sección.

\fill	\stretch{Número}
-------	------------------

\fill Es una longitud elástica cuyo valor es 0pt plus 1fill.

\stretch Es una longitud con valor natural 0pt y holgura un Número de unidades fill que puede ser entero o decimal. Así, \fill es equivalente a \stretch{1}. El comando puede ser utilizado, por ejemplo, para posicionar objetos dentro de una caja con una anchura fijada que excede a la suma de las longitudes de los objetos que contiene, distribuyendo el exceso de espacio disponible en la forma que convenga; a tal fin, entre cada pareja de objetos, se incluye este comando, eventualmente con diferentes valores de su argumento, para conseguir distribuir el exceso de espacio de forma proporcional a los valores de tales argumentos.

EJEMPLO 1.12

```
Comenzaremos... texto.\par\noindent
\vrule\hspace{\stretch{1}}
Centrado
\hspace{\stretch{1}}\vrule
\par Ahora colocamos... \par\noindent
\vrule\hspace{0pt plus 1fill}
Desplazado
\hspace{0pt plus 3fill}\vrule\par
O bien al revés\par\noindent
\vrule\hspace{\stretch{3}}Corrido%
\hspace{\stretch{1}}\vrul
```

Comenzaremos usando estas unidades de holgura para centrar un texto.

Centrado	Ahora colocamos un texto de tal modo que a su derecha haya el triple de espacio del que hay a su izquierda	Desplazado
		O bien al revés

		Corrido
--	--	---------

Distribución del espacio usando longitudes elásticas. Hemos utilizado el comando \vrule para imprimir la raya vertical de referencia; en general, en su lugar se utiliza \mbox{} o bien \null

Otros comandos interesantes, en relación con las elasticidades, son los que aparecen a continuación. Algunos de ellos, cuyo nombre resulta elocuente por sí mismo, se obtienen combinando

los comandos `\hspace` y `\vspace` con longitudes elásticas que hacen uso de las unidades comentadas anteriormente `fil` y `fill`.

<code>\hfil</code>	<code>\vfil</code>	<code>\rulefill</code>	<code>\downbracefill</code>
<code>\hfill</code>	<code>\vfill</code>	<code>\dotfill</code>	<code>\upbracefill</code>
<code>\hfilneg</code>	<code>\vfilneg</code>		<code>\leftarrowfill</code>
<code>\hss</code>	<code>\vss</code>		<code>\rightarrowfill</code>

Los comandos `\hfill` y `\vfill` ya fueron descritos en el apartado I.18.5, y son equivalentes, respectivamente, a `\hspace{\fill}` y `\vspace{\fill}`. Por otra parte, los comandos `\hfil` y `\vfil` son análogos a los anteriores, solamente que utilizan para la elasticidad una unidad `fil` en lugar de `fill`.

EJEMPLO 1.13

```
A \hfil B \hfil C\par
\f\parfillskip 0pt
A \hfil B \hfil C\par}
A \hfill B \hfill C\par
A \hfil B \dotfill C\par
A \rulefill B \hfil C
```

A	B	C
A	B	C
A	B	C
A B		C
A _____		B C

Comparación entre los efectos de los comandos `\hfil` y `\hfill`.

La primera línea del ejemplo anterior resulta engañosa, pues de acuerdo con la descripción que se ha realizado, cabría esperar que la ‘C’ tuviera la posición que tiene en la segunda línea; así debería ser, pero TEX antes de romper una línea (para evitar que las líneas cortas se estiren hacia la derecha) inserta el equivalente a un comando `\hfil` mediante el valor almacenado en la longitud `\parfillskip`, que se describe un poco después, lo que explica el resultado obtenido; en la segunda línea hemos puesto a cero el valor de dicha longitud a fin de conseguir el resultado esperado.

Los comandos de las dos últimas líneas, `\rulefill` y `\dotfill`, utilizan la misma tecnología de las longitudes elásticas, junto con un motivo (una raya, un punto, una flecha hacia la izquierda o hacia la derecha, una llave horizontal hacia abajo o hacia arriba) que se extiende hasta llenar el espacio disponible.

EJEMPLO 1.14

```
\centerline{Triángulos}\centerline{\downbracefill}
\centerline{%
Equiláteros\hfill Isósceles\hfill Escalenos}
```

The diagram shows three categories of triangles: Equilateral, Isosceles, and Scalene, arranged horizontally under a brace. The brace is positioned above the word "Triángulos".

Los comandos `\hfilneg` y `\vfilneg` equivalen básicamente a `\hspace{0pt plus -1fil}` y `\vspace{0pt plus -1fil}`, respectivamente, y permiten cancelar el efecto de los comandos `\hfil` y `\vfil`.

EJEMPLO 1.15

```
\parindent0pt \parfillskip0pt
\newcommand*\centrar[1]{%
\vrule\hfil #1\hfil\vrule }
\centrar{Centrado}\par
\centrar{Centrado anulado\hfilneg}\par
\centrar{\hfilneg Centrado anulado}
```

	Centrado	
	Centrado anulado	

El comando `\hss` es esencialmente equivalente a `\hspace{0pt plus 1fil minus 1fil}` y algo análogo ocurre con el comando `\vss`, reemplazando `\hspace` por `\vspace`. El comando `\hss` aparece en la definición de otros comandos de más alto nivel, como los comandos `\leftline`, `\rightline` y `\centerline`, que ya se introdujeron en el apartado I.3.3. También se emplea en las definiciones de los comandos

<code>\rlap{Objeto}</code>		<code>\llap{Objeto}</code>
----------------------------	--	----------------------------

que no detallamos aquí porque hacen uso de cajas de `TEX` más generales que las consideradas en la lección 18 (véase la sección 3.5.1), pero sí describiremos su efecto. En ambos casos *Objeto* se coloca en una caja de anchura 0 pt, con lo cual *Objeto* se sale de dicha caja; en el caso del primer comando lo hace por la derecha y para el segundo por la izquierda. Pero como para `LATEX` la caja tiene anchura cero, «la cabeza de impresión» no se ha movido, lo que puede provocar una sobreescritura.

EJEMPLO 1.16

```
Un... consume \rlap{espacio}\rule{.4pt}{2ex}\par
Y ... izquierda\llap{\rule[2.5pt]{33pt}{.4pt}}\par
que tampoco consume.\par
Fíjese bien:\hfil
\rlap{uno}\llap{dos}\vrule
\hfil ¿sorprendido? \par
¿Lo entiende ahora? \hfil
\llap{dos}\rlap{uno}\vrule
\hfil\null\par
\mbox{} \llap{Nos salimos} por la izquierda ...
```

Un texto hacia la derecha que no consume `\espacio`.
 Y ahora hacia la `izquierda` que tampoco consume.
 Fíjese bien: `\dos` `\uno` ¿sorprendido?
 ¿Lo entiende ahora? `\dos` `\uno`
 Nos salimos por la izquierda utilizando el comando `\llap` precedido
 del comando `\leavevmode` (o bien `\mbox`) para entrar en el
 modo horizontal (véase la sección 3.5.1). Obsérvese que las
 cuatro primeras líneas son más cortas a causa de la sangría.

El secreto de las unidades `fil`

Seguramente, con las descripciones realizadas y los ejemplos que tratan de ilustrar su funcionamiento, ha conseguido forjarse una idea sobre la unidad `fil`, pero no es menos probable que aún le resulten un tanto misteriosas.

Lo primero a señalar es que las unidades `fil` aparecen siempre en una caja con dimensiones establecidas; en ocasiones la caja puede estar implícita (la caja de la línea o la página, por ejemplo). Su función está relacionada con estiramientos de espacios (`plus`), cuando el material que contiene la caja no alcanza el tamaño de ésta, o «compresiones» de espacios (`minus`), cuando las dimensiones del material contenido en la caja superan las de ésta.

Para distribuir el espacio en una caja que contenga longitudes elásticas \TeX necesita asignar un valor concreto a la unidad `fil`, y esto lo realiza en dos etapas. Por sencillez en la descripción nos limitaremos al caso en que la caja tenga mayor longitud que su contenido, es decir, al caso en que el contenido de la caja tenga un déficit de longitud en relación con el tamaño de la misma; el otro caso es análogo, sólo que referido al exceso de tamaño del contenido en lugar de al déficit.

Etapa 1. Realiza la suma de las longitudes de todas las cajas contenidas en la caja en cuestión (incluyendo eventualmente los espacios rígidos si los hay), y si dicha longitud es menor que la longitud de la caja, hay un déficit de longitud, que llamaremos d . Como $d \geq 0$, se interesa únicamente de las unidades `plus` y realiza la suma de todos los `fil` que existen en la caja (cada uno con su signo), lo cual proporciona un valor, digamos n . El resultado del cociente $d/|n|$ es el valor que, en esa caja, tiene 1 `fil`.

Etapa 2. Va insertando entre las cajas contenidas en la caja de referencia las unidades `fil` que se indican, donde toman el valor de 1 `fil` el calculado en la etapa anterior, considerando en esta etapa únicamente el valor absoluto de las longitudes `fil`.

<code>\leftskip</code>	<code>\rightskip</code>	<code>\parfillskip</code>
------------------------	-------------------------	---------------------------

corresponden a longitudes que \TeX utiliza internamente. Los dos primeros se insertan en cada línea (adicionalmente al material que contiene) al principio y al final de la misma, respectivamente. Sus valores por defecto son 0 pt y son longitudes elásticas. Si se utilizan de forma explícita dentro de un grupo, es esencial finalizar el párrafo antes de cerrar el grupo.

El comando `\parfillskip` controla el espacio añadido automáticamente al finalizar el párrafo, después de suprimir todos los espacios existentes tras la última caja del mismo. Su valor por defecto es 0pt `plus 1fil`. Poniendo dicho valor a cero, en un párrafo relativamente largo, se consigue distribuir el espacio de tal manera que todas las líneas tengan la misma longitud, sin que el espacio entre palabras se estire demasiado.

Los valores de `\Leftskip`, `\rightskip` y `\parfillskip` pueden indicarse en cualquier lugar del párrafo; si se asignan dos valores distintos a una misma longitud, tendrá validez la última asignación. En el siguiente ejemplo vemos una aplicación (véase el ejemplo 8.10 en la pág. 473 para otra aplicación).

EJEMPLO 1.17

Continuando con el ejemplo anterior, éste sería un párrafo normal, en el que dichos márgenes están a cero cm.`\par {\leftskip=1cm \rightskip=5mm}` Pero ahora hemos construido un párrafo más estrecho (1cm por la izquierda y 5mm por la derecha) que nos permite realizar una cita o bien escribir algo importante.`\par}`

Continuando con el ejemplo anterior, éste sería un párrafo normal, en el que dichos márgenes están a cero cm.

Pero ahora hemos construido un párrafo más estrecho (1cm por la izquierda y 5mm por la derecha) que nos permite realizar una cita o bien escribir algo importante.

Utilizando estas longitudes de forma inteligente pueden conseguirse efectos tales como párrafos enteros sangrados por ambos lados (parecidos a los producidos por el entorno `quote`), párrafos con líneas centradas o «párrafos en triángulo español».

EJEMPLO 1.18

```
{\leftskip Opt plus 1fil  
{\rightskip Opt plus -1fil  
{\parfillskip Opt plus 2fil  
Con frecuencia la \'ultima l\'inea de un ...  
aparece centrada.\par}
```

Con frecuencia la \'ultima l\'inea de un p\'arrafo es m\'as corta que las anteriores. Habitualmente esa l\'inea aparece justificada a izquierda. En los llamados «p\'arrafos espa\'oles» aparece centrada.

P\'arrafos espa\'oles. Suelen utilizarse en leyendas de figuras y cuadros (v. la opci\'on centerlast del paquete caption2 descrito en la secci\'on 3.4.3)

La tradici\'on espa\'ola para presentar un verso que necesita dos l\'ineas es diferente de la que LATEX realiza con el entorno `verse` (v\'ease el apartado I.3.4). En nuestra tradici\'on, la segunda l\'inea aparece justificada a derecha. Utilizando las ideas del ejemplo 1.18 (m\'as concretamente cambiando `2fil` por `1fil`) no es dif\'icil construir un entorno similar a `verse` que produzca ese comportamiento.

PARA SABER M\'AS

-  Puede que, a lo largo de la lectura de las p\'ginas anteriores, se haya preguntado c\'omo los autores de este libro han conseguido escribir los comandos, sin que \'estos se ejecuten, es decir, c\'omo escribir en este punto '`\documentclass`' o '`\pagebreak`'. En el archivo `PieDeLaLetra.pdf` encontrará una descripci\'on de la escritura denominada «`verbatim`» (el comando `\verb` y el entorno `verbatim`), que tiene este efecto. Adem\'as este archivo contiene la descripci\'on de otros comandos, como `\obeylines`, `\obeyspaces`, `\ignorespaces` y `\unskip`.
-  En el archivo `MasSignos.pdf` encontrará todas las opciones del paquete `inputenc`, as\'i como la forma de introducir acentos y signos de manera totalmente independiente de la plataforma, es decir, sin utilizar `inputenc` y recurriendo \'unicamente al manejo de los 128 primeros caracteres ASCII, los compartidos por todas las p\'ginas de c\'odigos. \'Est\'a es la forma «pura», la original de TEX, de introducir estos signos.

Capítulo

2

Estructura

EN este capítulo se abordan diferentes temáticas, todas ellas de carácter global, es decir, que afectan a los documentos en su totalidad. Se completa la información sobre algunos temas ya tratados en la primera parte, por ejemplo, la cuestión de cómo gestionar la numeración de las unidades de estructura (sección 2.1) y el control de los índices general, de cuadros y de figuras (sección 2.2). Aparecen, sin embargo, cuestiones total o parcialmente nuevas: la posibilidad de componer un documento largo por fragmentos, una completa y rica descripción de los estilos de página que, como ya sabemos, marcan el aspecto general de un documento, y un buen montón de parámetros y opciones que determinan longitudes y comportamientos generales. Sin duda, son de la mayor importancia las dos últimas secciones, referidas, respectivamente, a índices alfabéticos y bibliografía. Los índices alfabéticos no pueden faltar en un libro técnico; por supuesto, tampoco la bibliografía y, aunque de ésta ya aprendió un poco en la lección 15, aquí mostramos cómo generarla mediante un proceso automático, procedimiento que también se utiliza en el caso de los citados índices alfabéticos.

2.1. La numeración de las unidades de estructura

EN el apartado I.6.2 fueron descritos los comandos disponibles para crear las unidades de estructura en las clases `article`, `book` y sus derivadas. Se afirmaba allí que, salvo para las versiones con asterisco de tales comandos, normalmente las unidades de estructura son numeradas automáticamente. Si decide hacer una experiencia por sí mismo creando un documento que contenga hasta las unidades de estructura de más bajo nivel, el resultado que obtendrá puede hacerle dudar de la certeza de tal afirmación. Además, si comprueba el índice general verá que no todas las unidades de estructura aparecen en él. A pesar de ello, todas las unidades de estructura (salvo las «estrelladas») son numeradas internamente; lo que ocurre es que no siempre se imprime el número que les corresponde precediendo a los respectivos títulos y tampoco aparecen todas en el índice general.

Entonces surgen de manera natural las siguientes cuestiones: ¿para qué unidades se imprime el número de orden?, ¿cuáles aparecen listadas en el índice general?

Para responder a estas preguntas comenzemos indicando que cada unidad de estructura posee un *nivel*: una sección siempre tiene nivel 1, las «subsecciones» nivel 2, las «subsubsecciones» nivel 3, y así sucesivamente. Las unidades jerárquicamente superiores a las anteriores tienen un

nivel distinto según la clase de documento: en `article` el nivel de las partes (iniciadas con `\part`) es 0, mientras que en `book` los capítulos tienen nivel 0 y las partes -1.

En L^AT_EX existen dos contadores que se refieren al *nivel* de las unidades estructurales:

<code>secnumdepth</code>	<code>tocdepth</code>
--------------------------	-----------------------

El valor de `secnumdepth` fija el nivel más bajo en la jerarquía cuyo número de orden será impreso. El valor de `tocdepth` fija el nivel más bajo en la jerarquía para el que se incluirá una línea en el índice general. La siguiente tabla muestra los valores por defecto de estos contadores:

	<code>article</code>	<code>book</code>
<code>secnumdepth</code>	3	2
<code>tocdepth</code>	3	2

La modificación de estos valores puede realizarse mediante los comandos `\setcounter` y `\addtocounter`, ya descritos en el apartado I.17.1. La modificación de `secnumdepth` tendrá efecto inmediato, mientras que una modificación de `tocdepth` tendrá efecto global y un buen sitio para incluirla es el preámbulo del documento.

Veamos ahora cómo numera L^AT_EX las diferentes unidades estructurales.

A cada unidad, introducida en nuestro documento mediante los comandos de estructura, le corresponde un contador de nombre igual al del comando que inició la unidad sin la barra de comando (por ejemplo, a `\chapter` le corresponde el contador `chapter` y a `\section` el contador `section`). Cada uno de estos contadores (véase la página 139) lleva asociado un comando que imprime su representación, el comando compuesto por `\the` y el nombre del contador (en los ejemplos anteriores `\thechapter` y `\thesection`).

Cuando se inicia una unidad de estructura ordinaria (es decir, no «estrellada»), cuyo nivel no es inferior al que fija `\secnumdepth`, el comando responsable de dicho inicio escribe el número de la unidad o, más precisamente, la representación del correspondiente contador, utilizando para ello, naturalmente, el comando `\the...` adecuado.

La representación más habitual de los contadores de las unidades de estructura está formada por el número de la unidad, en notación arábiga, precedido de los números de las unidades de estructura que son superiores a ella en la jerarquía y que, por tanto, la contienen, estando separados tales números entre sí mediante un punto. Esto es lo más usual; sin embargo, cuando la unidad es «muy profunda», es decir, muy baja en la jerarquía, esta idea suele cambiar, ya que mantenerla podría suponer una larga lista de números, lo que representaría, además de una falta de estética, una numeración poco práctica. Otra excepción la constituye la unidad `part`, cuya numeración no suele aparecer en la representación del contador de las unidades en ella contenidas. Para alterar la representación por defecto de estos contadores debemos utilizar los comandos expuestos en el apartado I.17.1¹.

¹Observe que, en general, en las referencias a elementos de una parte de este libro, realizadas desde la otra, hemos alterado la representación del contador `section` para que la unidad `part` figure en ella.

Finalmente, cuando se inicia una unidad de estructura ordinaria (por ejemplo, `\section`) se incrementa el contador correspondiente a su tipo de unidad (el contador `section`) y se ponen a cero todos los contadores de las unidades que le siguen en el orden jerárquico (`subsection`, `subsubsection`, etc.); es decir, el contador de cada tipo de unidad de estructura está subordinado al contador de la unidad inmediatamente por encima de ella en la jerarquía (y como consecuencia, se reinicia a cero cuando comienza una nueva unidad de un tipo superior cualquiera). Hay, sin embargo, una excepción a esta dinámica, referida también a las unidades de tipo `\part`: ningún contador de unidad de estructura está subordinado al contador `part`; por tanto, si, por ejemplo, una parte termina con el capítulo 5, el primer capítulo de la parte siguiente llevará el número 6.

En el apartado I.6.2 señalamos que, con la opción `spanish` de `babel`, en el caso de las unidades de estructura generadas con `\part` y `\chapter` el encabezamiento incluye, precediendo al número de la unidad, los antetítulos «Parte» y «Capítulo», respectivamente. Dichos antetítulos son reemplazados por otros al utilizar diferentes opciones de `babel`. Si no se utiliza el paquete `babel` los antetítulos que aparecen son «Part» y «Chapter», respectivamente.

En realidad el paquete `babel` cambia estos antetítulos mediante la simple redefinición de los comandos que almacenan el contenido de los mismos:

<code>\partname</code>	<code>\chaptername</code>
------------------------	---------------------------

Los valores asignados a estos comandos pueden ser personalizados en la forma habitual utilizando un `\renewcommand*` o, si se utiliza `babel` en la forma indicada en la página 169, a propósito de los comandos `\captionsIdioma`.

2.2. Más sobre índices: general, de cuadros y de figuras

Y a hemos descrito cómo obtener el índice general de un documento (apartado I.6.5), así como los índices de figuras (apartado I.9.4) y cuadros (apartado I.14.4). Para cada una de ellas, L^AT_EX utiliza un fichero auxiliar en el que se escriben las anotaciones necesarias para construir tales índices: en cada compilación el compilador escribe en los ficheros auxiliares la información que debe aparecer en los correspondientes índices, mientras que en la compilación siguiente utiliza dichos ficheros auxiliares para construir de hecho los índices, con el formato adecuado; volviendo a su vez a reescribir los ficheros auxiliares. Cada uno de estos ficheros auxiliares, que se generan únicamente con `\tableofcontents`, `\listoffigures` y `\listoftables`, respectivamente, reciben el mismo nombre que el fichero que se está compilando, y su extensión depende del tipo de índice, tal y como se refleja en la tabla siguiente:

Índice:	general	de figuras	de cuadros
Comando:	<code>\tableofcontents</code>	<code>\listoffigures</code>	<code>\listoftables</code>
Extensión:	<code>toc</code>	<code>lof</code>	<code>lot</code>

Cada uno de estos tres índices se inicia con un título, cuyo valor es el asignado por los respectivos comandos:

<code>\contentsname</code>	<code>\listfigurename</code>	<code>\listtablename</code>
----------------------------	------------------------------	-----------------------------

Por defecto, los valores de estos comandos son, en el mismo orden: «Contents», «List of Figures» y «List of Tables», valores que pueden ser cambiados por el paquete `babel` (véase la sección 1.2), o desde el texto fuente utilizando `\renewcommand*`. Internamente, los comandos de inicio de estas unidades utilizan un comando de estructura estrellado (`\chapter*` o `\section*`, según la clase de documento).

2.2.1. Inclusión manual de entradas

Cada uno de los comandos para crear unidades de estructura (`\part`, `\chapter`, `\section...`) se ocupa automáticamente de introducir información en el fichero de extensión `toc`, y el comando `\caption`, utilizable en los entornos `figure` y `table`, se ocupa de hacerlo, respectivamente, en los ficheros de extensión `lof` y `lot`. Conviene recordar (véase la sección 2.1) que no todos los comandos que crean unidades de estructura escriben en el fichero `toc`.

Pero además, podemos incluir, en cualquiera de los tres tipos de índice, los textos y comandos que deseemos, de forma que se tiene así un gran control sobre la información contenida en los tres índices. Veamos los dos comandos que desarrollan estas posibilidades.

`\addcontentsline{ExtensiónFichero}{Unidad}{TextoEntrada}`

Este comando añadirá (en la segunda compilación posterior a su inclusión) una línea o *entrada* al índice especificado por el argumento *ExtensiónFichero*. Los valores que *ExtensiónFichero* puede tomar son `toc` para añadir al índice general, `lof` para el índice de figuras y `lot` para el de cuadros.

El argumento *Unidad* especifica a qué tipo de unidad de estructura queremos asimilar dicha entrada. Valores admisibles pueden ser entonces: `part`, `chapter`, `section...`, `figure` o `table`. Cada *entrada* en estos índices tiene un formato predeterminado, distinto según la unidad a la que corresponde. Aunque la elección del argumento *Unidad* es independiente del valor de *ExtensiónFichero* (es posible, por ejemplo, incluir en el índice de figuras una línea diseñada como si correspondiera a una parte o a un capítulo), parece lógico que ambos argumentos se correspondan en su forma natural:

- Si *ExtensiónFichero* es `toc`, *Unidad* suele ser el nombre de un comando de estructura sin la barra de comando (por ejemplo: `chapter`, `section...`). Es importante hacer observar aquí que si la *Unidad* elegida en este argumento tiene un nivel que, de acuerdo con el valor actual de `tocdepth`, no sería incluida con el comando de estructura correspondiente, entonces `\addcontentsline` tampoco dará la entrada prevista. Asimismo el comando de estructura elegido debe estar disponible en la clase utilizada.
- Si *ExtensiónFichero* es `lof`, *Unidad* suele ser `figure`².
- Si *ExtensiónFichero* es `lot`, *Unidad* suele ser `table`².

El argumento *TextoEntrada* es lo que deseamos añadir en el índice correspondiente. Para facilitar la utilización de este argumento señalaremos, por ejemplo, que los comandos de estructura

²Si estamos utilizando el paquete `subfigure`, podemos dar el valor `subfigure` o `subtable` al argumento *Unidad*; estos valores corresponden a las entradas que los comandos `\subfigure` y `\subtable` incluyen en el índice de figuras o de cuadros respectivamente (véase el apartado PARA SABER MÁS del capítulo 3).

ra utilizan internamente el comando `\addcontentsline` y, como sabemos, para cada una de las líneas del índice se incluyen, por ese orden, la representación del contador correspondiente a la unidad de estructura, el título de la unidad y la página, pudiendo existir entre estos dos últimos una línea de puntos conductores (así ocurre siempre que la unidad especificada por el argumento *Unidad* no sea la primera ni la segunda en la jerarquía de la clase de documento en uso). El formato de las líneas depende del nivel de la unidad y de la clase de documento. De estos tres elementos que acabamos de mencionar, únicamente el primero y el segundo son incluidos mediante *TextoEntrada*; el número de página no forma parte de dicho argumento, sino que es recuperado internamente por el comando `\addcontentsline`.

El número inicial en una entrada se consigue mediante el uso de

`\numberline{Núm}`

Concretamente, para incluir el número *Núm* al inicio de la entrada, iniciaremos el argumento *TextoEntrada* con `\protect\numberline{Núm}` (el comando `\protect` es necesario, ya que `\numberline` es un comando frágil, véase la sección 2.2.2). De este modo podemos asignar números personalizados a las entradas incluidas mediante `\addcontentsline`.

El argumento *TextoEntrada* puede incluir comandos, por ejemplo, para dar formato a la entrada; en este caso, si *TextoEntrada* contiene comandos frágiles es necesario protegerlos con el comando `\protect` (véase la sección 2.2.2).

EJEMPLO 2.1

```
\tableofcontents
\section{Numeración de las unidades}
En este apartado aprenderemos los parámetros
que utiliza \LaTeX\ para numerar las...
\section{Más sobre índices}
\LaTeX\ genera por sí mismo el índice...
\subsection{Inclusión manual de leyendas}
Cada una de las unidades de estructura...
\addcontentsline{toc}{subsection}%
{\protect\numberline{2.1.a}Inclusión de textos}
\addtocontents{toc}{%
\medskip\noindent\bfseries\itshape
  Información adicional\\ Ejemplos}
```

Inclusión manual de leyendas en el índice general

Índice

1. Numeración de las unidades	1
2. Más sobre índices	5
2.1. Inclusión manual de entradas	6
2.1.a. Inclusión de textos	9

Información adicional
Ejemplos

Otro comando, similar a `\addcontentsline`, pero de más amplia intención es

`\addtocontents{ExtensiónFichero}{Objeto}`

con el que podemos añadir al índice que deseemos (especificado por *ExtensiónFichero*) cualquier *Objeto*, incluyendo en éste tantos comandos como sean necesarios. Con `\addtocontents` podemos escribir tantas líneas o entradas como queramos (mediante el uso de `\backslash\backslash`), y no sólo una, como era el caso para `\addcontentsline`.

2.2.2. Comandos frágiles y robustos en argumentos móviles

A veces, el argumento de algunos comandos no sirve sólo para ser incluido en el fragmento de texto correspondiente a la página que se está componiendo, sino que también se usa en otras partes del documento. Por ejemplo, los comandos de estructura (`\chapter`, `\section...`) sirven, cuando se utiliza la clase de documento `book`, para definir el contenido de las cabeceras de las páginas o para que L^AT_EX construya el índice general.

A estos argumentos que aparecen en distintas partes del documento se los denomina *argumentos móviles*.

En estos argumentos se pueden producir problemas cuando contienen comandos que pueden tener distinta función dependiendo de la parte del documento en la que se encuentran (los argumentos móviles pueden estar en intersecciones de grupos no anidados). Los comandos que pueden producir estos problemas se denominan *comandos frágiles*, mientras que los comandos que pueden viajar por el documento sin producir problemas se llaman *comandos robustos*.

Básicamente, se puede afirmar que todos los comandos que admiten argumentos opcionales son comandos frágiles y que la mayor parte de los entornos son frágiles; mientras que los comandos de selección de fuentes y tamaños, los comandos que asignan o declaran valores y la mayor parte de los comandos del modo matemático son robustos.

Para intentar evitar que un comando frágil que debe aparecer dentro de un argumento móvil ocasione problemas se puede utilizar el comando

```
\protect
```

delante del comando frágil. Por ejemplo, si se compila el documento fuente siguiente:

```
\documentclass{book}
\begin{document}
\tableofcontents
\chapter{Un título con \protect\footnote{pie de página.}} ...
\end{document}
```

se puede asignar un pie de página al título del capítulo, aunque, tal y como está escrito, este pie de página viajará con el título al índice general, generado por el comando `\tableofcontents`.

El comando `\protect` sólo actúa sobre el comando que le sigue, de modo que si en el argumento móvil aparecen varios comandos frágiles, cada uno de ellos debe estar protegido por su propio `\protect`. El comando `\protect` no debe de utilizarse nunca con los comandos de declaración y manipulación de los contadores y longitudes descritos en la lección 17 y en la sección 1.5.

Los comandos y entornos descritos a continuación poseen argumentos móviles.

- Todos los comandos con argumentos que pueden viajar al índice general, a una lista de figuras o cuadros o a un índice cualquiera. Por ejemplo, los de estructura. Cuando se utilizan argumentosopcionales con los comandos de estructura para describir el texto que debe viajar a los índices, estos argumentos serán los argumentos móviles.
- Los comandos de generación de cabeceras y pies de página (véase la sección 2.4).
- El comando `\thanks`.
- Los argumentosopcionales del comando `\bibitem` (véase la lección 15).

2.3. Compilación por trozos

A estas alturas hemos adquirido un poco de experiencia con L^AT_EX y conocemos con qué facilidad se deslizan los errores, complicando la compilación. Una primera forma de facilitar la depuración de errores es compilar frecuentemente, a medida que avanza la escritura del documento fuente. Imaginemos que estamos escribiendo un documento bastante largo, por ejemplo unas 100 páginas. Cuando ya hayamos escrito, compilado y corregido hasta la página 50, para compilar lo que sigue a ésta tendremos que volver a compilar una y otra vez las 50 primeras.

En resumen, ¿no sería ventajoso poder compilar el documento por trozos, para, una vez libres todos ellos de errores, realizar una compilación final global? Podemos imaginar otras situaciones donde sería útil dividir el código fuente, por ejemplo: cuando queramos que un mismo texto sea común a varios documentos; si alguien nos ha proporcionado, o nosotros mismos hemos escrito, una buena cantidad de nuevos comandos de L^AT_EX y no deseamos incluir este largo código en el preámbulo de todos los documentos; cuando hemos formado una buena colección de fichas técnicas, bibliográficas o problemas y desearíamos, en cualquier momento, poder incluir en un documento las que nos interesen de entre todas ellas,... L^AT_EX nos abre esta posibilidad en dos formas, cada una con sus ventajas e inconvenientes, pero ambas realmente simples y útiles.

La primera de estas formas se consigue mediante el comando:

```
\input{Fichero}
```

Cuando el compilador encuentra este comando va a leer el *Fichero* indicado en el argumento y continúa compilando dicho *Fichero*.

El uso más común de este comando es el siguiente: se escribe un documento principal, digamos *MiDoc.tex*, cuya primera línea empieza con `\documentclass` y cuya última línea termina con un `\end{document}`. Entre ambos comandos escribimos el código que sea necesario y, por ejemplo, `\input{MiDoc1.tex}`. Cuando el compilador lea la frase `\input{MiDoc1.tex}` buscará el fichero *MiDoc1.tex* y, si lo encuentra, compilará el contenido de dicho fichero. Al terminar seguirá compilando el fichero *MiDoc.tex*. Es decir, es como si la frase `\input{MiDoc1.tex}` fuera reemplazada por el contenido de *MiDoc1.tex*. Se comprende, pues, que *MiDoc1.tex* no debe contener un comando `\documentclass`, ni ningún otro que no pueda ser repetido en un mismo documento y que ya contenga *MiDoc.tex*.

En la sintaxis anterior, se pueden eliminar las llaves que encierran al argumento *Fichero*, dejando, naturalmente, un espacio entre `\input` y dicho argumento. El argumento *Fichero* debe ser un nombre válido de fichero en nuestro sistema operativo incluyendo la extensión, salvo que dicha extensión sea *tex*, en cuyo caso puede omitirse. Si el *Fichero* no está en el mismo directorio que el fichero principal, en el ejemplo anterior *MiDoc.tex*, *Fichero* debe contener el camino o «path» completo a fin de que pueda ser encontrado³.

³Recuerde que cuando el símbolo de directorio en el sistema operativo sea «\», este símbolo debe ser sustituido por «/». Una alternativa, que no necesita el camino, es que el fichero se encuentre en el directorio que nuestra instalación del compilador tenga definido como directorio de lectura de ficheros, declarado en la variable del sistema de nombre `TEXINPUT` o similar.

El uso práctico para compilar por trozos consiste en escribir las distintas partes en varios ficheros fuente distintos, cada uno de ellos incluido con `\input` en el fichero principal; después bastará anteponer un `%` a cualquiera de estos `\input` para evitar que se compile el fichero correspondiente. Además, cualquier fichero incluido en uno principal, mediante un comando `\input`, puede contener, a su vez, otros `\input`, y así sucesivamente.

La segunda forma de dividir el código fuente es mediante la combinación de los comandos

<code>\includeonly{Fichero1,Fichero2...}</code>	<code>\include{Fichero}</code>
---	--------------------------------

Los argumentos de estos comandos no deben contener la extensión de los nombres de ficheros, la cual siempre se supone que es `.tex`. La acción específica de `\include` es idéntica a la de `\input`: es como si la frase `\include{MiDoc1}` fuera sustituida por el contenido de `MiDoc1.tex`. Ahora bien, esta sustitución o «inserción» en el documento principal sólo se llevará a cabo si en el preámbulo de este último no figura el comando `\includeonly` o bien figura con `MiDoc1` en su argumento. Por ejemplo, la frase `\includeonly{MiDoc1,MiDoc4}` en el preámbulo de `MiDoc.tex` hará que sólo se incluyan los ficheros `MiDoc1.tex` y `MiDoc4.tex`. Si no aparece `\includeonly` en el preámbulo del documento principal, entonces todos los ficheros incluidos con `\include` serán procesados; por el contrario, si el preámbulo contiene el comando `\includeonly` con argumento vacío, es decir, `\includeonly{}`, ninguno de estos ficheros será tenido en cuenta.

Hay, además, una característica fundamental del comando `\include` que no tiene `\input` y que es la que le da valor frente a éste: cada fichero que sea compilado como parte de un documento principal mediante un `\include` genera su propio fichero de extensión `.aux`. Por ejemplo, supongamos que el fichero `MiDoc1.tex` figura en un `\include` del documento principal `MiDoc.tex` y que ya ha sido compilado, utilizando `\includeonly{MiDoc1}`. Existirán, a partir de entonces, los ficheros `MiDoc.aux` y `MiDoc1.aux`. Ahora queremos utilizar el comando `\include{MiDoc2}` para compilar `MiDoc2.tex`, pero sin compilar `MiDoc1.tex`; para ello escribiremos, en el preámbulo, la frase `\includeonly{MiDoc2}`. Pues bien, L^AT_EX utilizará, al compilar `MiDoc.tex`, toda la información que escribió en los ficheros `MiDoc.aux` y `MiDoc1.aux`, y, por tanto, `MiDoc2.tex` será procesado *como si* `MiDoc1` hubiera sido insertado: conocerá la numeración de las páginas, de los capítulos, secciones, teoremas, etc., las referencias bibliográficas, las etiquetas para referencias cruzadas, la información para construir el índice terminológico... Observemos que la información auxiliar corresponde a la última compilación del fichero actualmente no incluido.

Como era lógico esperar, aunque esto supone una desventaja, el comando `\include` siempre inicia una nueva página y, cuando termina su función, es decir, cuando todo el fichero incluido ha sido procesado, se inicia otra nueva página. En ambos saltos de página se «expulsan» los objetos flotantes que pudieran quedar pendientes, es decir, se ejecuta `\clearpage` (véase la página 69).

Otra desventaja del comando `\include` es que no puede ser encadenado, es decir, que un fichero incluido con `\include` no puede contener, a su vez, ningún `\include`.

Es fácil encontrar problemas con `\include` si en un fichero incluido con dicho comando, digamos `MiDoc1`, definimos algún comando, contador, etc., que deja una «traza» en `MiDoc1.aux` y, posteriormente, en una nueva compilación del documento principal, excluimos `MiDoc1.tex`. La razón de ello es clara: en la segunda compilación se lee `MiDoc1.aux` y en éste se encuentra una referencia a un contador (o similar) que, ahora, es desconocido. Eliminando de `MiDoc1.aux` la

referencia que conduce al error (o simplemente borrando este fichero) el problema desaparecerá, aunque, con ello, también desaparecerán las ventajas de `\include`. Es una buena práctica reunir todas las definiciones de contadores, longitudes, comandos y entornos en el documento principal.

El interés de la combinación de los comandos `\include` e `\includeonly` es enorme, naturalmente en la escritura de documentos largos y estructurados: tesis, libros, etc. Este libro no podría haber sido compuesto sin recurrir a esta herramienta. Queda el inconveniente de que `\include` inicia siempre una página, pero si, por ejemplo, cada fichero incluido con este comando corresponde exactamente a un capítulo (se inicia con `\chapter`), entonces el inconveniente desaparece, podemos compilar por trozos y no será necesario ni siquiera repaginar el documento.

Incluyendo en el preámbulo del documento la declaración

```
\listfiles
```

conseguiremos que, al finalizar la compilación, se escriba, tanto en pantalla como en el fichero auxiliar `log`, la lista de todos los ficheros que el compilador ha leído para conformar el documento completo, incluyendo ficheros correspondientes a clases de documento y sus opciones, paquetes, fuentes y, naturalmente, los insertados mediante `\input` o `\include`. Esta lista puede llegar a ser una lista muy larga...

2.4. El estilo de las páginas

En el apartado I.7.1 vimos los estilos de página más importantes usados por las clases básicas de L^AT_EX: `empty`, `headings` y `plain`. En esta sección añadiremos uno nuevo de nombre `myheadings` (véase la página 216). Cada clase de documento selecciona por sí misma un estilo de página que, obviamente, puede ser cambiado mediante el uso de las declaraciones `\pagestyle` y `\thispagestyle` que fueron descritas en la lección 7. Concretamente, la clase `book` selecciona por defecto el estilo `headings`, mientras que la clase `article` utiliza por defecto el estilo `plain`.

Aunque un estilo particular esté determinado, ya sea por defecto o por nuestra propia elección, algunos comandos de L^AT_EX pueden elegir otro para una página concreta (utilizando internamente el comando `\thispagestyle`); he aquí algunas situaciones específicas (algunas de las cuales ya fueron señaladas en el apartado I.7.1) en las que se presenta este comportamiento especial.

- Cualquier página creada con el entorno `titlepage` tiene estilo `empty` en todas las clases estándar de documento.
- La página del título creada con `\maketitle` utiliza el estilo `plain` en todas las clases de documento cuando está activada la opción `notitlepage`. Si, por el contrario, se ha activado la opción `titlepage` entonces `\maketitle` invoca internamente un entorno `titlepage` y, por tanto, el estilo de la página del título es `empty`.
- El entorno `abstract` no cambia el estilo de la página en el que se incluye, salvo que la opción `titlepage` esté activada, en cuyo caso dicho entorno invoca también un entorno `titlepage` y, por tanto, el mismo comentario anterior se aplica.
- En la clase `book` la página donde `\part` inicia la unidad «parte» tiene estilo `plain`. En la clase `article` el estilo `headings` se ve modificado de una forma especial, dejando en ésta

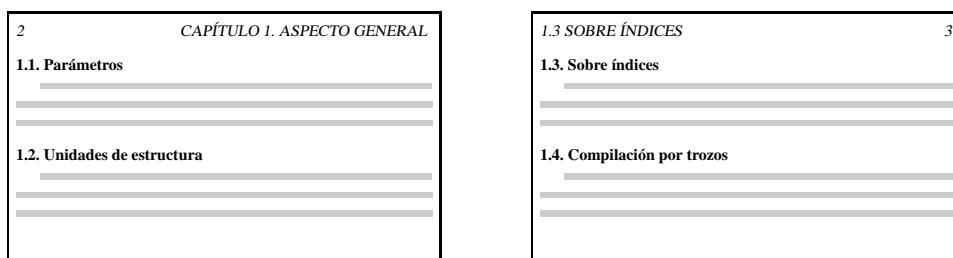


Figura 2.1: Marcas con estilo de página `headings` en la clase `book`

únicamente el número de página (concretamente, se ejecuta la sentencia `\markboth{}{}`, explicada más adelante).

- La página donde se inicia un capítulo (mediante `\chapter`) tiene estilo `plain`.

2.4.1. Las «marcas» de la cabecera y el pie

El estilo `headings` define el pie como vacío y en la cabecera incluye el número de página (en el borde externo de la misma), y además puede incluir un texto, que depende de la clase que esté en uso y de sus opciones; dicho textosuele denominarse «marca» para `LATEX`, y, en terminología tipográfica, «folio». Estas marcas de la cabecera contienen, generalmente, información obtenida de los argumentos de los comandos de estructura (o de modificaciones de los mismos realizadas mediante los comandos `\chaptermark`, `\sectionmark` y `\subsectionmark` descritos en el apartado I.6.2) y el número de la unidad de estructura.

Possiblemente una de las cuestiones más discutibles, y discutidas, de las múltiples decisiones que `LATEX` adopta al componer un documento, sea el formato utilizado por el estilo `headings`. El neófito pronto encuentra problemas cuando utiliza este estilo: las marcas se amontonan o superan el espacio disponible, a menos que los títulos de las unidades de estructura sean cortos. La utilización de los comandos `\chaptermark`, `\sectionmark` y `\subsectionmark` permite eludir los problemas; pero es únicamente un paliativo para solventar una dificultad que, en muchos casos, podría admitir una solución más natural, utilizando caracteres de menor tamaño en las marcas, o suprimiendo las mayúsculas, o el término ‘CAPÍTULO’ que aparece en la clase `book`... En esta sección profundizaremos en el conocimiento del modo en el que se asignan las marcas y aprenderemos, especialmente en el apartado 2.4.2, a personalizar dichas marcas. Según nuestra experiencia, únicamente mantienen el diseño estándar del estilo `headings` creado por L. Lamport quienes no han aprendido lo suficiente para cambiarlo.

Los comandos que contienen el valor de estas marcas para la cabecera de cada página a izquierda y a derecha son `\leftmark` y `\rightmark`, respectivamente. Estos comandos no deben ser redefinidos directamente, con la intención de modificar las marcas, ya que esto fijaría dicha definición hasta una nueva. Para modificar las asignaciones que, por defecto, realiza el estilo `headings` para las marcas disponemos de los comandos

`\markboth{EncabPágIzquierda}{EncabPágDerecha}`

`\markright{EncabPágDerecha}`

donde *EncabPágDerecha* es el valor asignado a la marca de las páginas a derecha (controladas por `\rightmark`), mientras que *EncabPágIzquierda* es el valor que tomará la marca de las páginas a izquierda (controladas por `\leftmark`). En el caso de utilizar la opción `oneside` el único comando relevante es `\markright`.

Parece lógico pensar que las marcas `\leftmark` y `\rightmark` contendrán como valores para las respectivas marcas los valores especificados por el último `\markboth` y/o el último `\markright` en una página; sin embargo, esto no es exactamente así. La razón de esta aparente pérdida del gran «sentido común» mostrado por L^AT_EX reside en el complicado algoritmo de fragmentación de páginas y en la, aún más compleja, «output routine» del compilador T_EX. Fundamentalmente el compilador no se detiene exactamente en el punto donde acaba la página que vemos impresa, sino que para llegar a componer ésta ha considerado una buena cantidad de texto adicional y sólo después de ello ha tomado la decisión sobre el punto más apropiado en el que cortar la página. Se produce así un cierto «asincronismo» en la división de páginas. Es fácil comprender entonces que si se tomara como valor activo siempre el último indicado, se podría tener para una página un valor de la marca que ha sido asignado en un párrafo que definitivamente figura en la página siguiente.

L^AT_EX opta entonces por el siguiente criterio, que genera sus propios inconvenientes⁴: el valor que toma `\leftmark` para una página es el especificado por el argumento *EncabPágIzquierda* del último `\markboth` en dicha página. El valor asignado en una página a `\rightmark` es el del argumento *EncabPágDerecha* del primer comando `\markright` o `\markboth` en dicha página. En cualquiera de los dos casos, si ninguno de estos dos comandos está presente en la página, entonces el valor será el último asignado en la página anterior (que, por tanto, no coincide necesariamente con el «activo» en dicha página anterior).

En todo caso, si en algún momento llega a verse sorprendido por las marcas generadas, debe tener en cuenta que la asignación de valores a los parámetros `\leftmark` y `\rightmark` la realiza la «output routine» de L^AT_EX y que, por tanto, sólo cobra valor al finalizar la página. Los comentarios realizados en la lección 6, a propósito del comportamiento no esperado de los comandos `\sectionmark` y `\subsectionmark` en determinadas situaciones, y la forma de proceder, se aplican igualmente al comando `\markright` ya que, como veremos a continuación, existe una relación directa entre estos comandos.

Es interesante comprender cómo realizan los comandos de estructura la asignación de las «marcas», porque, una vez comprendido el mecanismo utilizado estaremos en condiciones de modificar, de forma sencilla, el comportamiento que, por defecto, realiza L^AT_EX. Esto permitirá, al mismo tiempo, ilustrar el manejo de los comandos `\markboth` y `\markright`.

⁴En la clase `article` con estilo `headings` y opción `twoside` las marcas de la página a izquierda son generadas por los comandos `\section`, mientras que las marcas de las páginas a derecha son generadas por los comandos `\subsection`. Imaginemos un documento con secciones muy cortas de forma que en una página a izquierda aparecen: sección 1, sección 1.1, sección 1.2, sección 2. Y la sección 2 se prolonga a lo largo de toda la página siguiente (a derecha). De acuerdo con el algoritmo utilizado por L^AT_EX, en la página a izquierda aparecerá el título de la sección 2, mientras que en la página a derecha aparecerá el título de la subsección 1.1, lo cual resulta lamentable. En situaciones singulares como ésta es necesario introducir manualmente un comando `\sectionmark` o `\subsectionmark` en el lugar oportuno.

Los comandos `\chaptermark`, `\sectionmark` y `\subsectionmark` son definidos en las diferentes clases de documento en el momento de definir el estilo `headings` y el valor de `Texto` en dicha definición es el *Título* de la unidad correspondiente o el *TextoToc*, en el sentido indicado en el apartado I.6.2. son definidos en las diferentes clases de documento en el momento de definir el estilo `headings` y el valor de `Texto` en dicha definición es el *Título* de la unidad correspondiente o el *TextoToc*, en el sentido indicado en el apartado I.6.2. Cada vez que se inicia una unidad de estructura se ejecuta uno de tales comandos. En la clase `book` con opción `twoside` (opción por defecto) los comandos `\chaptermark` y `\sectionmark` son los encargados de asignar las marcas en las páginas a izquierda y a derecha, respectivamente; en la clase `article` dichas asignaciones se realizan, respectivamente, con los comandos `\sectionmark` y `\subsectionmark`, aunque su efecto no podrá ser percibido en esta clase a menos que se utilice de forma explícita el estilo `headings`.

El cuadro 2.1 recoge las *definiciones simplificadas* de los comandos `\dotsmark` que son significativos para las clases `book` y `article`. Por ejemplo, el análisis del código que, para la clase `book` con la opción `twoside`, figura en dicho cuadro es muy interesante por los dos motivos que siguen.

1. Pone de relieve que si bien los comandos `\markboth` y `\markright` son los responsables de guardar información sobre las marcas, que utilizan `\leftmark` y `\rightmark`, en la práctica, en el estilo `headings`, son los comandos de estructura (sin asterisco) los que, a través del comando correspondiente `\dotsmark`, proporcionan esa información para la mayor parte de las «páginas ordinarias» (cuando un documento se inicia las marcas están vacías y los índices y otras unidades de estructura estrelladas pueden hacer su propio uso de `\markboth` y `\markright`). Cualquier asignación directa de las marcas que realice el autor con `\markboth` y `\markright` únicamente tendrá efecto hasta la siguiente unidad de estructura del nivel adecuado.
2. Explica el formato adoptado por las marcas en el estilo `headings`. Concretamente, al definir `\chaptermark` se utiliza un comando `\markboth` en cuyo primer argumento pone en letras mayúsculas (`\MakeUppercase`) lo siguiente: el valor de `\chaptername` (véase la sección 2.2), un espacio de separación, la representación del contador `chapter` seguido de un punto y de un espacio y finalmente el valor `#1` (que antes hemos denotado como `Texto` y que, recordemos, correspondía al argumento obligatorio u optativo del comando `\chapter`). En cambio, su segundo argumento es vacío. La definición del comando `\sectionmark`, que puede ser fácilmente interpretada por similitud, utiliza el comando `\markright`. Como se observa, el número de la página (guardado en el contador `page`) y su posición es gestionado internamente por L^AT_EX.

Junto al comando `\MakeUppercase` que sirve para convertir `Texto` a letras mayúsculas, existe un comando dual que lo convierte a letras minúsculas.

<code>\MakeUppercase{Texto}</code>	<code>\MakeLowercase{Texto}</code>
------------------------------------	------------------------------------

A la vista del código anterior resulta ahora sencillo modificar el formato de las marcas para, por ejemplo, suprimir las mayúsculas, el comando `\chaptername` y el perfil inclinado (introducido por comandos internos de más bajo nivel) y disminuir el tamaño de los caracteres usando además versalita. El código sería:

Marcas en páginas a izquierda y a derecha en la clase book	
twoside	<pre>\renewcommand*{\chaptermark}[1] {% \markboth{% \MakeUppercase{\chaptername} \ thechapter. \ #1}{}} \newcommand*{\sectiononmark}[1] {% \markright{% \MakeUppercase{\thesection. \ #1}}}</pre>
oneside	<pre>\renewcommand*{\chaptermark}[1] {% \markright{% \MakeUppercase{\chaptername} \ thechapter. \ #1}}}</pre>
Marcas en páginas a izquierda y a derecha en la clase article	
twoside	<pre>\renewcommand*{\sectiononmark}[1] {% \markboth{% \MakeUppercase{\thesection\quad \#1}--\#1}{}} \newcommand*{\subsectionmark}[1] {% \markright{% \MakeUppercase{\thesubsection\quad \#1}}}</pre>
oneside	<pre>\renewcommand*{\sectiononmark}[1] {% \markright{% \MakeUppercase{\thesection\quad \#1}}}</pre>

Cuadro 2.1: Definiciones de las marcas en el estilo headings. Con opción oneside todas las páginas son «a derecha»

```
\renewcommand*{\chaptermark}[1]{%
    \markboth{\small\upshape\scfamily\thechapter.\ #1}{}}%
\renewcommand*{\sectionmark}[1]{%
    \markright{\small\upshape\scfamily\thesection.\ #1}}%
```

Es conveniente poner estas definiciones en el preámbulo o, en el caso de que se utilice el código `\pagestyle{headings}`, después de éste.

Puesto que los comandos que inician unidades de estructura con «estrella» no modifican el valor previo de las marcas, si deseamos llevar a cabo esta tarea, podemos incluir un `\markboth` o `\markright` dentro de su argumento; por ejemplo:

```
\chapter*{Prólogo\markboth{Prólogo}{Prólogo}}
```

El estilo myheadings

Siempre que la estructura básica de las cabeceras se conserve, las ideas anteriores permiten modificar fácilmente el formato de las marcas proporcionadas por el estilo `headings` (para modificaciones más profundas es aconsejable además utilizar el paquete `fancyhdr` que describimos en el apartado siguiente); en particular sería posible fijar unas marcas personales, únicas o cambiantes a lo largo del documento, que no fueran modificadas por los comandos de estructura.

`LATEX` tiene definido un estilo de página, de nombre `myheadings`, que hace esa tarea por nosotros. En dicho estilo, el pie es vacío y la cabecera contiene únicamente el número de página, en la parte más externa de la misma, permaneciendo las marcas vacías, lo que nos obliga a incluir un `\markboth` o un `\markright` cada vez que queramos especificar o cambiar el contenido de la cabecera. Este estilo puede resultar de utilidad, por ejemplo, para incluir una misma cabecera a lo largo de todo el documento, o de una parte del mismo.

Páginas blancas sin cabecera La configuración inicial de la clase `book` usa la opción `openright` (véase el apartado I.6.6) que hace que todos los capítulos se inicien en página a derecha. Esto provoca que, en ocasiones, la página a izquierda inmediatamente anterior al inicio del capítulo no contenga nada, salvo la cabecera. Algunas personas preferirían que esas páginas fueran completamente blancas.

Una forma de conseguir ese objetivo es incluir manualmente, cuantas veces sea necesario, tras la línea que finaliza la página a izquierda inmediatamente anterior a la página en blanco, el código `\clearpage{\thispagestyle{empty}\cleardoublepage}`

También es posible conseguir que esta operación se realice de forma automática al comienzo de cada capítulo mediante el siguiente código introducido en el preámbulo del documento:

```
\makeatletter
\renewcommand*{\cleardoublepage}{\clearpage\if@twoside
    \ifodd\c@page\else
        \hbox{}\thispagestyle{empty}\newpage
    \if@twocolumn\hbox{}\newpage\fi\fi\fi}
\makeatother
```

Se puede observar que esta redefinición (además de contener una parte de código posiblemente incomprensible para el lector y que no explicaremos) necesita una precaución muy particular, que hasta ahora no hemos encontrado. La razón de esta particularidad reside en el signo @ que aparece en el nombre de algunos de los comandos utilizados. Los comandos cuyo nombre contenga este carácter sólo pueden ser utilizados, en principio, en las clases de documento o en los paquetes; la idea original que justifica esta restricción es la siguiente: debido a la profusión de comandos internos que se manejan en los paquetes, podría darse la casualidad de que el nombre de un comando definido en un paquete coincidiera con el nombre de un comando definido por el usuario; puesto que obviamente esta coincidencia ha de ser evitada, el uso del carácter @ en el nombre de los comandos de los paquetes alcanza ese objetivo. Por otro lado, dado que este carácter es inexistente en el lenguaje ordinario, es poco probable que alguien desee definir un comando con un nombre que lo contenga y, además, al estar prohibido su uso en un documento, se evita que el usuario pueda redefinir comandos internos de las clases y paquetes.

Para poder utilizar en un documento L^AT_EX comandos cuyo nombre contiene el carácter @ debemos utilizar los comandos

`\makeatletter`

`\makeatother`

El primero debe preceder al bloque de código donde se utilicen comandos con @ y el segundo seguirlo. Como los propios nombres indican, el primero convierte a @ en una letra más, mientras que el segundo lo vuelve a convertir en «otra cosa».

2.4.2. Estilos mejorados: el paquete fancyhdr

A pesar de que las posibilidades que ofrecen los cuatro estilos de página de L^AT_EX son bastante amplias, ninguna de ellas permite variaciones en el aspecto visual de los encabezamientos o pies de página. Por ejemplo, no es fácil incluir una raya horizontal que separe las cabeceras y los pies del cuerpo de la página, como tampoco lo es que figure en el pie de todas las páginas un texto fijo.

El paquete fancyhdr, desarrollado por Piet Van Oostrum [50], proporciona herramientas muy sencillas para construir cabeceras y pies de página con un alto grado de personalización. La estructura de las cabeceras y pies que el paquete proporciona, representada en la figura 2.2, consta de tres partes (izquierda, central y derecha), tanto para la cabecera como para el pie, y sendas rayas de separación con el cuerpo de la página. La cabecera y el pie pueden ser determinados de forma independiente para las páginas a izquierda y a derecha.

Pero, además, cada una de las tres partes citadas puede constar de varias líneas (de hecho, cada una constituye una caja) y la anchura total de la cabecera y el pie puede ser asignada explícitamente. Naturalmente cualquiera de estos elementos es opcional.

Para poder usar el estilo de página que proporciona el paquete fancyhdr es preciso, además del correspondiente `\usepackage`, incluir la declaración que especifica fancy como el estilo de página a utilizar y que es:

`\pagestyle{fancy}`

La declaración anterior asigna internamente el valor de `\textwidth`, la anchura del texto, a una nueva longitud:

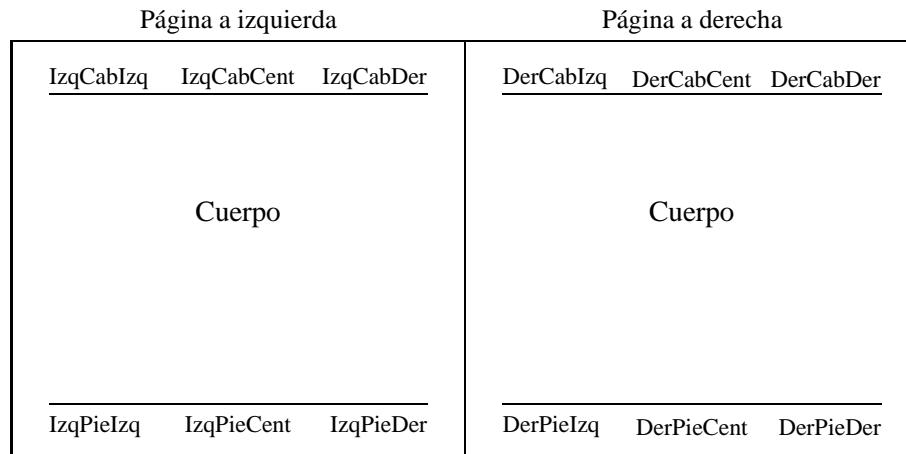


Figura 2.2: Esquema de la cabecera y el pie de las páginas a derecha e izquierda definidos por el paquete fancyhdr

\headwidth

que es la anchura de las cajas que contendrán la cabecera y el pie. La longitud \headwidth puede ser modificada directa e independientemente en cualquier parte del documento fuente. Es recomendable que la modificación de esta longitud figure, en el texto fuente, después de la declaración \pagestyle{fancy}. Por ejemplo, el siguiente código construye una cabecera y un pie de página que se extiende horizontalmente hasta incluir, además del texto usual, las notas al margen, para lo cual se añade a la longitud \headwidth la suma de las longitudes correspondientes a la distancia que separa las notas al margen del texto principal y la anchura de la caja destinada a contener las notas al margen.

```
\pagestyle{fancy}
\addtolength{\headwidth}{\marginparsep}
\addtolength{\headwidth}{\marginparwidth}
```

Los siguientes comandos son los encargados de especificar cada uno de los elementos en la cabecera («header») y el pie («foot»):

\fancyhead [Selectores] {Cabecera}

\fancyfoot [Selectores] {Pie}

Los argumentos *Cabecera* y *Pie* deben contener los textos que quieran ser incluidos en la cabecera y pie de página, respectivamente. El argumento opcional *Selectores* consiste en una lista de caracteres o combinaciones de dos caracteres, separados por comas si se incluyen varios grupos, y cuyo objetivo es el de especificar a qué páginas y a qué partes, de las tres en que se dividen la cabecera y el pie, se aplican los argumentos *Cabecera* y *Pie*. Los caracteres que forman los *Selectores* se indican en el cuadro 2.2; así, por ejemplo, [L] como argumento opcional de \fancyhead hará que *Cabecera* determine la parte izquierda de la cabecera de todas las páginas (sean a derecha o a izquierda), [LE,RO] aplicará el argumento correspondiente a la parte izquierda de las páginas a izquierda y a la parte derecha de las páginas a derecha. Los caracteres pueden escribirse tanto

en minúscula como en mayúscula y pueden incluirse grupos. Naturalmente, si estamos utilizando la opción `oneside` el selector correspondiente a las páginas a izquierda [E] es ignorado y, aunque esté presente, la definición de la cabecera y el pie será la misma para todas las páginas.

Selectores de página	E	Página a izquierda (Even, par)
	O	Página a derecha (Odd, impar)
Selectores de campo	L	Parte izquierda
	C	Parte central
	R	Parte derecha
Selectores de cabecera o pie para \fancyhf	H	Cabecera
	F	Pie

Cuadro 2.2: Selectores de página y campo en el paquete fancyhdr

El paquete `fancyhdr` proporciona un nuevo comando, más general que los anteriores y sobre el que aquéllos están basados:

`\fancyhf [Selectores] {Texto}`

Este comando puede ser utilizado para determinar tanto la cabecera como el pie, incluyendo en su argumento opcional *Selectores* un nuevo carácter, indicado en el último bloque del cuadro 2.2. Así, el comando `\fancyhf` admite *Selectores* formados por uno, dos o tres caracteres, cada uno de ellos tomado de uno de los bloques horizontales de dicho cuadro. Si en todos los *Selectores* se incluye el carácter ‘H’, entonces el comando es equivalente a `\fancyhead` con los mismos *Selectores* (privados de dicho carácter), y, naturalmente, algo análogo ocurre con el carácter ‘F’ y el comando `\fancyfoot`.

Es conveniente tener en cuenta las siguientes precauciones:

- si se incluye un *Selector* que sólo contiene el carácter ‘H’ (respectivamente, ‘F’), entonces *Texto* será el valor asignado a *las tres partes* de la cabecera (respectivamente, del pie), obteniéndose un resultado bastante poco agradable;
- si se especifican dos valores distintos para alguna parte de la cabecera o pie de las mismas páginas, el valor que prevalecerá será el último asignado. Así, por ejemplo, el código
`\fancyhf [FC] {\thepage}\fancyfoot [CO] {Capítulo \thechapter}`
dejaría el número de página en el centro del pie de las páginas a izquierda y el número del capítulo, precedido de «Capítulo», en el centro del pie de las páginas a derecha.

Los argumentos obligatorios de estos comandos (`\fancyhead`, `\fancyfoot` y `\fancyhf`) admiten en su interior saltos de línea (\backslash) y saltos verticales mayores implementados mediante `\vspace` o `\backslash [Salto]`, pudiendo construir así cabeceras o pies de varias líneas. Sin embargo `fancyhdr` no se responsabiliza, en ningún caso, de aumentar la altura que L^AT_EX destina a las cabeceras y pies; se hace preciso, pues, que nosotros modifiquemos las longitudes `\headheight`, `\headsep` y `\footskip` en la forma apropiada (véase la sección 2.5).

El estilo `fancy` incluye una raya horizontal debajo de la cabecera y otra encima del pie; ambas separan dichos elementos del cuerpo de la página. El grosor de dichas rayas viene determinado por los siguientes comandos:

<code>\headrulewidth</code>	<code>\footrulewidth</code>
-----------------------------	-----------------------------

que pueden ser modificados con un `\renewcommand*`, por ejemplo, en la forma siguiente

```
\renewcommand*{\footrulewidth}{0,5pt}
```

Los valores que `fancyhdr` asigna a dichos comandos son 0,4 pt para `\headrulewidth` y 0 pt para `\footrulewidth` (es ya conocido que un grosor de 0 pt hace invisible la raya).

Los comandos responsables de imprimir estas rayas son, respectivamente,

<code>\headrule</code>	<code>\footrule</code>
------------------------	------------------------

que pueden ser redefinidos para cambiar el aspecto de dichas rayas; por ejemplo, para una línea con puntos conductores el autor del paquete propone en [50]:

```
\renewcommand*{\headrule}{\vbox to 0pt{\hbox to \headwidth{\dotfill}\vss}}
```

La distancia vertical entre esta raya del pie de página y la parte superior del texto del pie propiamente dicho la controla el comando

<code>\footruleskip</code>

El paquete `fancyhdr`, con la elección del estilo `fancy`, asigna valores por defecto a los distintos campos de la cabecera y pie. La definición es

```
\fancyhead[LE,RO]{\slshape \rightmark}
\fancyhead[LO,RE]{\slshape \leftmark}
\fancyfoot[C]{\thepage}
```

y el resultado se muestra en la figura 2.3. Como se observa, a diferencia de lo que ocurre con los estilos `headings` y `myheadings`, con el estilo `fancy` el autor es también responsable de imprimir el número de página en la posición que desee.

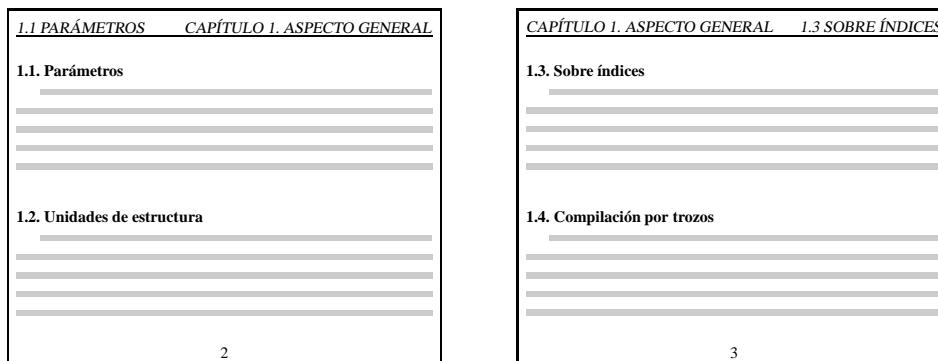


Figura 2.3: Marcas con el paquete `fancyhdr` y el estilo `fancy`. Los registros `\leftmark` y `\rightmark` se utilizan independientemente de que la página sea a izquierda o derecha. El valor de estos registros lo generan, habitualmente, los comandos de estructura, en la forma descrita en la sección 2.4.1

Si queremos cambiar estos valores iniciales, es una buena práctica iniciar dichos cambios con la sentencia `\fancyhf{}`, que vacía todas las asignaciones realizadas previamente. Además, como ya indicamos en el apartado I.2.4.1, se puede influir sobre el modo en el que se muestra en la cabecera la información de capítulos, secciones y subsecciones (¡sólo dos de ellos!). Basta para ello con redefinir los comandos `\chaptermark`, `\sectionmark` y `\subsectionmark`. Tales redefiniciones deben ser realizadas después de la primera llamada a `\pagestyle{fancy}`. Conviene advertir que redefinir los comandos `\chaptermark`, `\sectionmark`... puede no eliminar todas las mayúsculas de las marcas debido a que algunas de las marcas pueden haber sido generadas no con uno de estos comandos sino de forma directa usando `\markboth` o `\markright` mediante alguna construcción similar a las que aparecen en el cuadro 2.1 que incorporan el comando `\MakeUppercase`; tal como ocurre, por ejemplo, con los comandos `\tableofcontents`, `\listoffigures`, `\listoftables` y los entornos `thebibliography` y `theindex`. Una solución contundente es redefinir el comando `\MakeUppercase` para que no haga nada, pero eso es una solución drástica que puede tener contraindicaciones si el comando `\MakeUppercase` es utilizado en algún otro lugar en el documento.

El paquete `fancyhdr` aporta otra solución que no tiene las contraindicaciones que acabamos de señalar. A tal fin implemente el comando

```
\nouppercase
```

que, incluido en los argumentos de los comandos `\fancyhead`, `\fancyfoot` y `\fancyhf`, anula el efecto del comando `\MakeUppercase` en esos ámbitos (debe ser incluido con cuidado si en las marcas aparecen números romanos en minúscula, lo cual no es adecuado en español).

Definiendo y redefiniendo estilos de página con `fancyhdr`

Como ya sabemos, en ocasiones `LATEX` cambia internamente el estilo de página que el usuario ha seleccionado; por ejemplo, algunos comandos de estructura (como `\chapter` o `\part`) ejecutan la declaración `\thispagestyle{plain}`.

El paquete `fancyhdr` nos permite redefinir cualquier estilo de página existente (como `plain`), de forma que, incluso, podemos controlar esas páginas especiales. Además `fancyhdr` permite definir, de forma muy cómoda, nuevos estilos de página, aunque siempre restringiéndonos al uso de los comandos propios del paquete, anteriormente expuestos. El comando que realiza esta tarea es:

```
\fancypagestyle{Estilo}{Definición}
```

en el que el argumento *Estilo* es el nombre del estilo de página que va a ser definido (o redefinido); puede ser, pues, tanto un nombre de estilo nuevo, como uno de los ya definidos. El argumento *Definición* debe contener comandos de `LATEX` o del paquete `fancyhdr` que cambian las cabeceras y/o pies de página, incluyendo, si se desea, redefiniciones de los comandos `\headrulewidth` y `\footrulewidth`.

Por ejemplo, el código siguiente recupera los valores por defecto de los estilos `fancy`, `plain` y `empty`, añadiendo un texto fijo (aquí, figuradamente, «El título de mi libro») en la parte externa del pie de todas las páginas:

```
\fancyfoot[ro,le]{El título de mi libro}
\fancypagestyle{plain}{\fancyhf{}\renewcommand*\headrulewidth{0pt}%
  \fancyfoot[C]{\thepage}\fancyfoot[Ro,Le]{El título de mi libro}}
\fancypagestyle{empty}{\fancyhf{}\renewcommand*\headrulewidth{0pt}%
  \fancyfoot[r0,1E]{El título de mi libro}}
\pagestyle{fancy}
```

Observemos que tanto la redefinición `\headrulewidth` como la sentencia `\fancyhf{}` han sido incluidas en las redefiniciones de `plain` y `empty`; esto es necesario ya que el efecto inicial de `\fancypagestyle{Estilo}` es el de identificar el estilo `fancy` a *Estilo*. Un efecto secundario de este procedimiento es que, en las dos redefiniciones, debemos incluir la sentencia `\fancyfoot[ro,le]{...}`.

Otras utilidades

Piet van Oostrum ha definido otros comandos que permiten un control extremadamente fino sobre los estilos de página, algunos de ellos están incluidos en versiones recientes de `fancyhdr` y otros en el paquete `extramarks` que se incluye en la distribución estándar de `fancyhdr`. También se incluye el paquete `lastpage`, que escribe en el fichero `Midoc.aux` una etiqueta de nombre `LastPage` que guarda el número de la última página, permitiendo acceder a dicho número a través de `\pageref{LastPage}`. De ese modo hace posible, por ejemplo, incluir textos del tipo «página 9 de 56» mediante el código «`página~\thepage\` de `\pageref{LastPage}`».

En la documentación que acompaña al paquete `fancyhdr` se indica también el procedimiento a usar para construir diccionarios, conforme a la costumbre de incluir en la cabecera de la página la primera, o la última, entrada que aparece en dicha página.

Los tres comandos que siguen, incluidos en `fancyhdr`, permiten cambiar el estilo de las páginas que contienen elementos flotantes:

```
\iffloatpage{ValorFlota}{ValorNormal}
\iftopfloat{ValorFlota}{ValorNormal}
\ifbotfloat{ValorFlota}{ValorNormal}
```

Estos comandos pueden ser incluidos en cada uno de los argumentos que determinan cabeceras y pies de los comandos de `fancyhdr` (`\fancyhead`, `\fancyfoot`, `\fancyhf`), así como en el argumento del comando `\renewcommand*` que redifina la anchura de las rayas de cabecera y/o pie. Con `ValorFlota` especificamos el valor que tomará la correspondiente marca de cabecera o pie para una página que contenga elementos flotantes, mientras que `ValorNormal` será el valor especificado para las restantes páginas. Téngase en cuenta que no todas las páginas con elementos flotantes son afectadas por estos comandos; únicamente lo serán aquéllas que el compilador añada para incluir los objetos flotantes pendientes de ubicar, siempre que se haya incluido el comando `\iffloatpage`, o las que posean un objeto flotante en la parte superior o inferior de la página, controladas respectivamente por `\iftopfloat` e `\ifbotfloat`. Obsérvese que las acciones que realizan estos comandos no son, en absoluto, sencillas; debido a la propia naturaleza de los objetos flotantes, no es posible determinar el estilo de las páginas que los contengan con, digamos por ejemplo, un `\thispagestyle`.

El paquete `extramarks` [50] proporciona nuevos comandos de control sobre las marcas. Respecto a las marcas usuales disponemos de los comandos

`\firstleftmark`

`\lastrightmark`

con los que se pueden recuperar, respectivamente, el primer valor asignado a la marca izquierda y el último asignado a la marca derecha en una página. A estos valores se aplican los comentarios ya realizados anteriormente: la asignación de valores se lleva a cabo sólo al terminar la página. Estos comandos complementan, pues, las acciones de L^AT_EX, y pueden recuperar para su uso en las cabeceras o pies el primer y último valor de cada marca asignados en cada página.

Otros comandos introducidos por `extramarks` son:

`\extramarks{M1}{M2}`

`\firstxmark`

`\lastxmark`

El primero de ellos proporciona dos nuevas marcas con valores *M1* y *M2*. Los otros dos recuperan, respectivamente, el primero de los *M1* y el último de los *M2* asignados en la página.

Con el fin de explicar los posibles usos de estas «marcas extras», tomamos un ejemplo de [50]: supongamos que en algunos párrafos que pueden ser partidos por un salto de página deseamos incluir un texto del tipo «(Continúa en la página siguiente...)» al final del primer fragmento y «(Continuación...)» al comienzo del segundo fragmento ubicado en la siguiente página. Naturalmente deseamos automatizar el proceso, de forma que tales textos no aparezcan si el párrafo termina siendo incluido en una sola página. Introducimos para ello el siguiente código, que puede ser transformado fácilmente en un entorno:

```
\extramarks{}{(Continúa en la página siguiente\...)}
Este párrafo puede ser partido por un salto de página...
\extramarks{(Continuación\...)}{}
```

en el que «Este párrafo...» representa nuestro texto. Combinando entonces el código anterior con el estilo `fancy` en la forma:

```
\fancyhead[LE]{\firstxmark}\fancyfoot[RO]{\lastxmark}
```

obtendremos el resultado pretendido. La explicación es sencilla: si el párrafo no se divide entre dos páginas consecutivas, los valores tomados por las marcas extras son ambos vacíos; por el contrario, si el párrafo se ve cortado, para la página en la que se inicia dicho párrafo, la cabecera a izquierda toma el valor vacío y el pie a derecha toma el valor de la última marca extra, es decir, «(Continúa en la...)»; en la página siguiente, por el contrario, esta marca se hace vacía y la parte izquierda de la cabecera toma el valor de la primera marca extra, es decir, «(Continuación...)». Para automatizar el proceso, puede definirse en siguiente entorno:

```
\newenvironment*{ParrafoEspecial}
{\extramarks{}{(Continúa en la página siguiente\...)}}
{\extramarks{(Continuación\...)}{}}
```

Con el fin de que los valores dados a las marcas extras no permanezcan para las siguientes páginas, es importante añadir a continuación de `\fancyfoot[RO]{\lastxmark}` un nuevo comando `\extramarks{}{}`.

2.5. Parámetros de composición de páginas

EN la lección 7 describíamos ya algunos parámetros básicos que controlan el tamaño y la posición de la mancha de texto. Concretamente se estudiaron la anchura y altura del cuerpo del texto (`\textwidth` y `\textheight`), los bordes de impresión izquierdo y superior (`\hoffset` y `\voffset`) y el margen izquierdo de las páginas a izquierda y a derecha (`\evensidemargin` y `\oddsidemargin`). En esta sección completamos esta información con la descripción de un buen número de parámetros adicionales, importantes a la hora de determinar el aspecto general de las páginas; nos referimos no al contenido de las distintas partes (folios, pies, etc.) sino a los espacios reservados para ellas, incluyendo cabecera y pie, notas al margen y al pie, etc. Muchos de ellos pueden ser visualizados en la figura I.7.1, página 66.

\columnwidth Es una longitud, calculada internamente por L^AT_EX, que mide la anchura de cada columna, cuando se está escribiendo a varias columnas. Su valor depende de `\textwidth`, del número de columnas y de la longitud `\columnsep` (véase la página 102). Naturalmente, coincide con `\textwidth` cuando se escribe a una sola columna. De forma natural, `\columnwidth` es también la anchura de la caja en la que se escriben las notas al pie de página y las leyendas de los entornos `figure` y `table`, cuando se escribe con la opción `twocolumn` de la clase de documento (o con `\twocolumn`). No ocurre así con las leyendas de los entornos `figure*` y `table*` (véase la figura 9.1, para el caso del entorno `figure*` y algo similar ocurre con el entorno `table*` mencionado en la lección 14), ni con las notas a pie en el entorno `multicols`, ya que ambas se colocan a todo el ancho de la mancha. No es en absoluto recomendable modificar directamente esta longitud, especialmente si se escribe a más de una columna.

\footnoterule Es el comando responsable de imprimir la marca de separación entre el texto ordinario y las notas a pie de página. En las principales clases de documento este comando coloca un filete de longitud 4/10 de la longitud `\columnwidth`. Pero podemos cambiar dicha marca en cualquier momento, redefiniendo el comando. Por ejemplo,

```
\renewcommand*{\footnoterule}{\mbox{}\hrulefill}
\renewcommand*{\footnoterule}{\mbox{}\dotfill}
```

producen, respectivamente, una línea continua y una línea de puntos, ambas de anchura igual a `\columnwidth`. L^AT_EX espera que el espacio vertical total ocupado por la marca que imprime `\footnoterule` sea nulo y, por tanto, cuando se redefine este comando conviene llevar a cabo las tareas necesarias para alcanzar esa altura nula, comenzando, por ejemplo, con un espacio vertical negativo; las redefiniciones anteriores podrían mejorarse como sigue:

```
\renewcommand*{\footnoterule}{%
  \vspace*{-3pt}\mbox{}\hrulefill\null\vspace*{2,6pt}}
\renewcommand*{\footnoterule}{%
  \vspace*{-3pt}\mbox{}\dotfill\null\vspace*{1,9447pt}}
```

(la altura del filete y de los puntos generados por `\hrulefill` y `\dotfill` son 0,4 pt y 1,0553 pt, respectivamente).

- \footnotesep** Es la separación vertical entre la base de la última línea de una nota a pie de página y la base de la primera línea de la siguiente nota y puede ser cambiada en cualquier momento.
- \footskip** Es la separación vertical entre las bases de la última línea del cuerpo de la página y la última del pie de la página. Una modificación del mismo desplaza verticalmente dicho pie (recuerde que las notas a pie de página no forman parte del pie de la página).
- \headheight** Es la altura de la caja que contiene la cabecera de las páginas.
- \headsep** Es una separación adicional entre la base de la cabecera de la página y la parte superior del cuerpo de la página.
- \hsize** Es una longitud específica de L^AT_EX. Normalmente coincide con \linewidth.
- \ linewidth** La anchura de la línea de texto en uso, es decir aquélla que se está formando en cada momento. Suele coincidir con \columnwidth, aunque dentro de los entornos que cambian los márgenes (por ejemplo, enumerate, description, verse, [...] , \parbox, minipage, etc.) sus valores pueden ser distintos. Una modificación directa no afectará al texto ordinario, aunque sí a los citados entornos que sigan a la modificación.
- \marginparpush** Cada nota al margen empieza en su lugar natural (alineada horizontalmente con la línea que contiene el comando \marginpar); por tanto, podría ocurrir que, de ser colocadas en dicho lugar, dos notas consecutivas se superpusieran. El compilador tiene en cuenta esta eventualidad y desplaza verticalmente y hacia abajo la segunda nota, evitando la superposición. El desplazamiento vertical adicional es \marginparpush. Las notas al margen son ubicadas al terminar de procesar el párrafo en el que se encuentran; por tanto, el último valor asignado a \marginparpush en un párrafo será el valor en uso para todas las notas al margen del mismo párrafo y de los siguientes (hasta una nueva asignación).
- \marginparsep** La separación horizontal entre las notas al margen y el cuerpo del texto ordinario. Puede ser redefinido para cada nota al margen (a diferencia de \marginparpush).
- \marginparwidth** La anchura de la caja que contiene una nota al margen. Se puede redefinir para cada nota dentro de un párrafo.
- \paperheight** La altura del papel en el que se va a imprimir. Son muchos los parámetros y longitudes que se determinan en función de esta longitud (y de \paperwidth). Es preciso indicar que las opciones de tamaño del papel para \documentclass asignan el valor apropiado a \paperheight y la mayor parte de las distancias que L^AT_EX maneja son calculadas en función de dicho valor. Por tanto, nuevas asignaciones de esta longitud, ya sea en el preámbulo o en el propio documento, no conseguirán cambiar los valores de dichas longitudes dependientes (a este respecto véase, en el apartado 2.6, la forma de imprimir en apaisado).
- \paperwidth** La anchura del papel en el que se va a imprimir. Se le aplican los mismos comentarios que a \paperheight.
- \pdfpageheight** La altura de la página cuando un documento es compilado con PDFL^AT_EX. Una de las ventajas de los documentos PDF es que cada página puede tener unas dimensiones propias, y PDFL^AT_EX nos proporciona este parámetro (y \pdfpagewidth) para aprovechar

esta ventaja. Naturalmente esta variación en el tamaño de las páginas sólo tiene sentido en situaciones muy particulares que, desde luego, no incluyen un documento previsto para ser impreso. Por ejemplo, los autores hemos utilizado esta posibilidad en un documento electrónico en el que cada página contenía una única imagen, asignando a dicha página las dimensiones de la imagen; al visualizar el documento a pantalla completa se obtenía un efecto de fondo negro, en el que la imagen destaca convenientemente. Es preciso, además, tener en cuenta que alterar esta longitud supone tener que modificar adecuadamente una buena parte de los restantes parámetros, porque PDFLATEX no los cambia por sí mismo.

\pdfpagewidth La anchura de la página cuando un documento es compilado con PDFLATEX. Se aplican los mismos comentarios que en el caso anterior (véase el apartado 2.6).

\skip\footins Es un comando de TeX que define el espacio vertical entre el texto principal y el comienzo de las notas a pie de página. Se trata, a pesar de su nombre, de una longitud elástica y, como tal, puede ser modificada. Su valor en las clases estándar es el equivalente del espacio introducido por \bigskip.

\topmargin La distancia vertical entre el «borde superior de impresión» (véase el apartado I.7.2) y la parte superior de la cabecera. Su valor puede ser modificado en cualquier momento.

\topskip La altura de la primera línea de texto en cada página. Esta primera línea tiene una altura distinta al resto. Aumentar dicha altura tiene el mismo efecto visual que incrementar la longitud \headsep.

Existen algunos paquetes que pueden ser herramientas útiles a la hora de estudiar y/o modificar los parámetros anteriores. Unos simplemente se ocupan de fijar algunos de estos parámetros, y no nos ocuparemos de ellos. Otros permiten obtener los valores de algunas longitudes, verificar ciertos comportamientos o tener una idea gráfica de los distintos parámetros de la página; algunos de este segundo grupo se citan a continuación.

layout diseñado por Kent McPherson, con adaptaciones posteriores de Johannes Braams [44].

Introduce el comando \layout mediante el cual, con una pequeña modificación de los autores de este libro, se realizaron los esquemas de la figura I.7.1; como se observa, pues, es de utilidad para alcanzar un diseño propio de las páginas.

pageframe de Cameron Smith [58], que permite obtener las marcas de guillotina, marcos impresos para las distintas partes de una página, así como una cuadrícula personalizada para medir distancias. Funciona con las clases estándar y, dado que su objetivo fundamental es el diseño de libros, sólo funciona correctamente en documentos con la opción twoside.

vmargin de Volker Kuhlmann, proporciona comandos para especificar de forma cómoda y rápida una buena parte de los parámetros aquí incluidos.

También son interesantes los documentos **testpage.tex** y **testa4.tex**. El primero de ellos, de Leslie Lamport y Stephen Gildea, con adaptación posterior de Rainer Schöpf, permite comprobar cuáles son los «bordes de impresión» que por defecto maneja un visor. El segundo, del último de los autores citados, permite la verificación de la escritura en papel DINA4.

2.6. Más opciones para las clases de documento

La mayor parte de las opciones disponibles para las clases de documento estándar han sido descritas en el apartado I.6.6. Únicamente hay que añadir tres opciones más: una relacionada con las fórmulas matemáticas (que se detalla en la sección 5.2), otra con la bibliografía (openbib, véase el ejemplo 2.16 en la página 255) y la opción `landscape` que abordamos en el siguiente apartado. Asimismo incluimos en esta sección más aspectos relacionados con la opción `draft`.

2.6.1. Imprimir en apaisado: la opción `landscape`

La opción `landscape` (apaisado) es una de las disponibles en las clases de documento usuales: `book`, `article` y sus variantes.

Por el nombre que tiene podría pensarse que con cargar dicha opción nuestro documento quedaría listo para que se pueda ver e imprimir en formato apaisado; pero la situación es un poco más compleja. En primer lugar hemos de señalar que lo único que hace dicha opción es permutar las dos dimensiones del papel, que están guardadas en sendas longitudes de nombres `\paperwidth` (anchura de papel) y `\paperheight` (altura del papel), véase la sección 2.5. Así, por ejemplo, si utilizamos la opción `a4paper` en la clase de documento, dichas longitudes son `\paperheight=297mm` y `\paperwidth=210mm`, lo que significa que L^AT_EX, por defecto, está suponiendo que se va a utilizar papel DINA4 con la orientación habitual. Utilizar la opción `landscape` tendría el efecto de declarar `\paperheight=210mm` y `\paperwidth=297mm`.

Ese cambio en las dimensiones del papel es significativo, porque varias longitudes que L^AT_EX calcula internamente dependen de los valores de `\paperwidth` y `\paperheight`, pero resulta insuficiente, en general, para poder ver e imprimir documentos en formato apaisado. En lo que sigue describiremos las actuaciones complementarias que deben realizarse para conseguir el objetivo perseguido y, a fin de facilitar la comprensión de las variables que intervienen, en principio nos limitaremos a abordar la cuestión en el marco de las clases usuales con su configuración estándar; posteriormente abordaremos un caso más general.

Supongamos que tenemos un documento, bien construido, con la opción `onecolumn` (es la opción por defecto en las clases estándar, véase I.6.6) y decidimos utilizar la opción `landscape` y compilarlo de nuevo. Tras la nueva compilación el resultado obtenido es un documento en el que la altura de la mancha disminuye de acuerdo con la altura disponible en el papel, pero la anchura de la mancha no se incrementa en correspondencia con la nueva anchura del papel; ¡afortunadamente!, porque en caso contrario el resultado estético hubiera sido deplorable. Asimismo, la mancha se desplaza en sentido horizontal para aparecer centrada en el papel.

Si además de la opción `landscape` incluimos en el documento la opción `twocolumn`, el resultado que se obtiene es un documento a dos columnas cuya mancha realiza una ocupación razonable del espacio disponible conforme a las dimensiones del papel. La diferencia con la situación anterior es que, ahora, al existir dos columnas el ancho de línea en cada columna no supera el número de caracteres en una misma línea que L^AT_EX considera estéticamente admisible.

Aunque los efectos de la opción `landscape` sean los que acabamos de describir, puede suceder que le resulte imposible ver e imprimir el documento resultante, dependiendo de cuál sea el

programa utilizado. Por ejemplo, con YAP o KDVI podrá comprobar lo que acabamos de describir, aunque para ello pueda ser necesario configurar adecuadamente estos programas, lo cual conseguirá realizar sin dificultad a través de los menús desplegables; en cambio, no podrá hacer lo mismo con ACROBAT READER o GHOSTVIEW a través de los menús desplegables de estos programas.

A continuación vamos a indicar el proceso que es necesario realizar para conseguir formatos PS y PDF correctos. En ambos casos es cuestión de configurar el tamaño de papel de forma adecuada a estos formatos, ya que la opción `landscape` no realiza por sí misma esa tarea.

Tamaños de papel y DVIPS

En general, DVIPS asume que el tamaño de papel es DINA4 y que la orientación del mismo es vertical. Pero también tiene definidos otros tamaños a los que se accede mediante el parámetro optativo `-t` (véase el fichero `OpcionesDvips.pdf`) al llamar al programa. Dicho parámetro optativo permite asimismo solicitarle el formato apaisado mediante el valor `landscape`. Así,

`dvips -t landscape NombreFichero`

producirá el resultado deseado en el fichero `NombreFichero.ps`, siempre que el tamaño de papel sea DINA4; de ser otro el tamaño de papel, por ejemplo, DINA3 (420 mm × 297 mm), habría que utilizar el siguiente código

`dvips -t a3 -t landscape NombreFichero`

El fichero `NombreFichero.ps` obtenido utilizando este procedimiento se puede ver e imprimir con GHOSTVIEW, sin las dificultades antes señaladas.

Existe también otra manera de informar a DVIPS de que estamos usando un formato apaisado, ya que, como hemos visto, la opción `landscape` sólo afecta al modo en el que L^AT_EX compone el documento. Esta vía alternativa tiene el atractivo de que aparece escrita en el documento fuente y no requiere usar parámetros optativos al utilizar DVIPS. Este procedimiento hace uso del comando `\special` (véase la página 331).

`\special{papersize=Ancho,Alto}`

siendo `Ancho` y `Alto` las dimensiones del papel. El comando debe aparecer en cualquier lugar en la primera página del documento, y un buen lugar para colocarlo es el preámbulo del documento. Eventualmente puede ser necesario modificar los valores de `\hoffset` y `\voffset` para ajustar la mancha. Insistimos en que este comando es un complemento a la opción `landscape` (o a las asignaciones de valores a los comandos `\paperwidth` y `\paperheight`), y no una alternativa a la misma. El procedimiento funciona también con DVIPDFM para producir un archivo PDF.

Documentos apaisados personalizados La descripción anterior es de aplicación, como señalamos oportunamente, a la configuración por defecto. Pero si queremos hacer un póster en formato apaisado con caracteres de «gran tamaño» o bien un tríptico, esa configuración puede no resultar adecuada. Documentos de ese tipo requieren la utilización de las longitudes `\textwidth`, `\hoffset`, etc., para diseñar la página de acuerdo con nuestras necesidades; pero aun así las ideas anteriores pueden ser aplicadas con éxito, como puede apreciarse en el ejemplo 2.2.

EJEMPLO 2.2

```
\documentclass%
[11pt,a4paper,landscape]{article}
\usepackage[latin1]{inputenc}
\usepackage[spanish]{babel}
\usepackage{%
    graphicx,multicol,times,marvosym}
\textwidth=27cm      \textheight=18cm
\hoffset=-7.5cm     \voffset=-3cm
\columnsep=1.5cm     \pagestyle{empty}
\special{papersize=297mm,210mm}
\begin{document}      \large
\begin{multicols}{3}
\null\vfill
\begin{itemize} \small
\item El curso será impartido ...
\end{itemize}
\vfill \columnbreak \null\vfill
\begin{center} \Huge\bfseries
El editor científico\ \ \ TeX ...
\vfill \columnbreak
\Large\rotatebox{90}{%
\begin{minipage}{\textheight}
\baselineskip=9mm
\null\vfill ....\end{minipage}}
\end{center}
\end{multicols}
\end{document}
```

Construcción de un tríptico usando la opción `landscape`. Varios de los parámetros del estilo de página han de ser cambiados y `DVIPS`, `DVIPDFM` y `PDFLATEX` han de ser informados del cambio en las dimensiones de la página

Tamaños de papel y PDFLATEX

Al igual que ocurre con `DVIPS`, `PDFLATEX` dispone de un fichero de configuración que determina el tamaño de papel usado por defecto, que suele ser DINA4. También con este programa podemos utilizar el fichero fuente para informar a `PDFLATEX` de las dimensiones del papel a través de las longitudes `\pdfpagewidth` y `\pdfpageheight` (véase la sección 2.5). Así, por ejemplo, si estamos empleando un papel DINA4, además de utilizar la opción `landscape` en la clase de documento debemos incluir en el documento fuente las declaraciones

```
\pdfpagewidth 297mm
\pdfpageheight 210mm
```

para conseguir el resultado esperado.

2.6.2. Algo más sobre la opción `draft` y los mensajes `Overfull`

La opción `draft` fue comentada en la lección 6. Recordemos ahora que su función principal (algunos paquetes le atribuyen otras) es la de marcar, con un pequeño rectángulo negro en el margen derecho, aquellas líneas cuya longitud sobrepasa, aunque sea de forma imperceptible, el

- El curso `El editor científico` se imparte en la Universidad de Murcia.
- Coordinador del curso: José Manuel Merino.
- Inscripción en el Departamento de Matemáticas y Ciencias de la Universidad de Murcia, Facultad de Ciencias, Campus de Espinardo, bulevar de la Encarnación s/n, 30100 Murcia (España).
- El plazo de inscripción es del 1 de octubre de 2001 al 1 de noviembre de 2001.
- No se establece límite de tiempo de inscripción.
- El curso `El editor científico` se imparte los jueves de 14 a 18 de cada mes.
- Destinado a estudiantes de la Universidad de Murcia y a titulados universitarios.
- Los contenidos de los cuatro plazos de estudio de la asignatura son los siguientes:

 - La introducción a `TeX` y a su manejo.
 - La introducción a `LaTeX` y a su manejo.
 - A profundizar en la configuración de `TeX`.
 - A profundizar en la configuración de `LaTeX`.

- Se considera impresario de quien realiza el curso programar intervenciones debidamente regulares.
- Para más información:

 - Departamento de Matemáticas.
 - Tfno.: 968 653199
 - <http://www.dmat.um.es>

El editor científico

TeX

CURSO DE PROMOCIÓN

EDUCATIVA

(PRESencial o INTERNET)



UNIVERSIDAD DE MURCIA

DEPARTAMENTO DE MATEMÁTICAS

.....

HOJA DE INSCRIPCIÓN EN EL CURSO:

El editor científico TeX

NOMBRE:

DIRECCIÓN:

CP Y CIUDAD:



ESTUDIOS:

valor `\textwidth`, con el fin de hacerlas fácilmente detectables. El grosor de este rectángulo viene determinado por la longitud

`\overfullrule`

que puede ser modificada en cualquier momento.

Parece también interesante señalar aquí que la longitud

`\hfuzz`

representa el margen de confianza para los mensajes de `Overfull` horizontal: los textos que sobresalgan una longitud inferior al valor de `\hfuzz` no serán catalogados con este «horrible calificativo». El valor por defecto de `\hfuzz` es 0,1 pt (inapreciable al ojo humano) y puede ser redefinido en cualquier momento. Las modificaciones son locales y tienen valor sólo mientras se está dentro del grupo en el que se han incluido; en particular, es posible modificar esta longitud para una caja y sólo para ella.

Aunque no se refiere a la opción `draft`, la longitud

`\vfuzz`

es análoga a `\hfuzz`, pero referida al modo vertical: controla el margen de confianza para los mensajes «`Overfull \vbox...`».

La división silábica es un trabajo duro y no exento de fallos. Si excepcionalmente todo falla, todavía nos queda un recurso; eso sí, un recurso que causa un cierto daño.

`\sloppy`

`\fussy`

La declaración `\sloppy` cambia la «mentalidad» del compilador, de forma que, a partir de ella, se muestra muy relajado a la hora de introducir «glues», es decir, le importa muy poco estirar cuanto sea necesario el espacio elástico entre dos palabras. Esto hace que casi ninguna línea produzca un mensaje de `overfull`, aunque muchas líneas producirán un mensaje del tipo:

`Underfull \hbox (badness 10000) in paragraph at lines 118--122`

Las palabras se seguirán dividiendo cuando sea posible, pero aquéllas que el compilador no sepa tratar pasarán a la línea siguiente produciendo, probablemente, una línea con espacios entre palabras mayores que los habituales.

La declaración `\fussy` anula la acción de `\sloppy` y no es preciso incluirla si no es con dicha intención, ya que es la declaración activa por defecto. También puede delimitarse la influencia de `\sloppy` en la forma ordinaria: encerrándola en un grupo. En cada párrafo el algoritmo de rotura de líneas depende de la declaración que esté activa al terminar el párrafo; por tanto, será la última de las declaraciones `\sloppy` y `\fussy` incluida en el párrafo la que tenga efecto.

`\begin{sloppypar}`
Texto
`\end{sloppypar}`

Es un entorno equivalente a

`\par{\sloppy Texto\par}`

Se observa, por tanto, que `\begin{sloppypar}` inicia un grupo y un nuevo párrafo, activando la declaración `\sloppy`, mientras que `\end{sloppypar}` termina un párrafo y el grupo, cerrando pues el campo de acción de la declaración `\sloppy`. Cuando la declaración `\sloppy` está activa, el valor por defecto de `\hfuzz` y `\vfuzz` es de 0,5 pt.

2.7. Índices terminológicos

El índice general de un libro resulta de suma utilidad para consultar la información que contiene el mismo; allí están recogidos los títulos de los capítulos, secciones, subsecciones, etc. Sin embargo, este índice no suele ser suficiente; con la mayor parte de los textos de envergadura, libros o informes, resulta casi imprescindible disponer de un índice terminológico o índice de materias, ordenados alfabéticamente, donde se pueda buscar una palabra y saber la página donde está el concepto correspondiente y en qué páginas se ha tratado. La elaboración manual de un índice terminológico es una tarea tediosa para el autor, más cuanto mayor sea la longitud del documento. ¿Y si se hace un cambio en la paginación? En este caso no sirve de nada el trabajo realizado y es necesario rehacer prácticamente todo el índice terminológico. Con la ayuda de un computador el trabajo puede ser más llevadero, en función del compilador o procesador de textos que se esté utilizando. Con L^AT_EX la tarea es todavía relativamente pesada, pero en combinación con el programa MAKEINDEX construir este tipo de índices es cosa de «coser y cantar».

L^AT_EX no produce un índice terminológico automáticamente tal y como produce el índice general, sino que se limita a incluir en un fichero auxiliar las palabras (o, más generalmente, términos) que le indiquemos que deben aparecer en el índice, junto con el número de página en la que hemos realizado la indicación. Naturalmente, si indicamos dos veces el mismo término, éste aparecerá también dos veces en el fichero auxiliar, con los correspondientes números de página de cada una de las apariciones del término indicadas.

Pero L^AT_EX no ordena alfabéticamente esta lista, probablemente enorme, de términos, ni tampoco se ocupa de componerla en una forma correcta: en dos o más columnas, en bandera por la derecha, etc. De estas dos tareas se encarga el programa MAKEINDEX.

Antes de entrar en la descripción técnica del proceso de generación de los índices terminológicos conviene hacer observar que realizar unos buenos índices, realmente útiles para el lector, no es fácil. L^AT_EX y MAKEINDEX nos pueden ayudar a generarlos, pero su contenido sólo puede ser decidido por nosotros. Es tentador generar los índices terminológicos a la vez que se escribe el documento. Piénselo dos veces y resista a la tentación. Es virtualmente imposible obtener alguna consistencia en un índice generado de este modo. Como dice Leslie Lamport [36],

Un índice persigue ayudar al lector a encontrar lo que está buscando. Con esto en mente, el sentido común puede ayudar a planificar qué debe estar en el índice y cómo debería organizarse. Como es a menudo difícil distinguir el sentido común de lo que no lo es, el consejo profesional es útil. [...] Puede ser fácil elegir qué palabras son importantes y mecánicamente generar un índice con una cita para cada una de las apariciones de estas palabras. Un índice como éste no será tan útil para el lector como uno preparado con más cuidado.

2.7.1. El proceso manual en breve

El comando de L^AT_EX para indicar que un término debe aparecer en el índice terminológico es

```
\index{Entrada!SubEntrada!SubSubEntrada}
```

que puede ser utilizado en cualquier lugar de nuestro documento. Con este comando indicamos que el texto *Entrada* debe aparecer en el índice. Los argumentos *SubEntrada* y *SubSubEntrada* son optativos y permiten introducir en el índice términos de segundo y tercer nivel. Para producir estas entradas el comando \index debe contener necesariamente la *Entrada* principal y, en su caso, la *SubEntrada* o *SubSubEntrada* separadas por el carácter !.

Cada aparición de \index escribe la entrada correspondiente en el fichero auxiliar ya citado, cuyo nombre será el del documento que estemos compilando y cuya extensión es *idx*. Es necesario precisar que sólo se incluirán las entradas correspondientes a los \index de las partes realmente compiladas, de modo que si un archivo es incluido mediante \include, pero no compilado (porque su nombre no figura en \includeonly) los \index que contenga no enviarán las entradas al archivo *idx* (este inconveniente desaparece con el paquete index, apartado 2.7.7). El ejemplo 2.3 muestra el tipo de contenido del fichero *idx*, que tratado posteriormente permite producir un índice terminológico como el que se muestra en el ejemplo 2.4. Observe que cada comando \index se transforma en un comando \indexentry que recibe dos argumentos: el primero es el mismo argumento de \index, y el segundo es el número de la página donde éste se ha insertado; el usuario no tiene por qué manejar directamente el comando \indexentry.

EJEMPLO 2.3

<pre>\makeindex \begin{document} [Pág. 2]...y Mendeléiev\index{Mendeleiev} [Pág. 6] El elemento sodio\index{sodio} puede obtenerse fácilmente... [Pág. 10] En las últimas décadas se han descubierto metales\index{metales} nuevos. [Pág. 15] El ejemplo más sorprendente es el aluminio\index{metales!aluminio} que [Pág. 18] Si hablamos del aluminio\index{metales!aluminio!densidad}</pre>	<i>MiDoc.idx</i> <pre>\indexentry{Mendeleiev}{2} \indexentry{sodio}{6} \indexentry{metales}{10} \indexentry{metales!aluminio}{15} \indexentry{metales!aluminio!densidad}{18}</pre>
--	--

Ficheros *idx*; entre corchetes se indica figuradamente el número de página en que se encuentra el texto

EJEMPLO 2.4

```
\usepackage[spanish]{babel}
\begin{document}...
\begin{theindex}
    \item Mendeléiev, 2
    \item metales, 10
        \subitem aluminio, 15
            \subsubitem densidad, 18
    \item sodio, 6
\end{theindex}
```

Entorno *theindex*

Índice alfabético

Mendeléiev, 2
metales, 10
aluminio, 15
densidad, 18
sodio, 6

El proceso manual de construcción del índice requeriría entonces tomar la información del fichero `idx` y procesarla. Esto supone ordenarla alfabéticamente y adaptar la sintaxis, todo ello, insistimos, ¡manualmente!, a la del entorno fundamental de L^AT_EX en la construcción de estos índices. Este entorno es

```
\begin{theindex}
\item Entrada, NúmeroPágina
  \subitem SubEntrada, NúmeroPágina
    \subsubitem SubSubEntrada, NúmeroPágina
...
\end{theindex}
```

que da formato en dos columnas a las entradas que han sido proporcionadas en los correspondientes `\item`, `\subitem`, `\subsubitem`; éstos corresponden, respectivamente, a entradas en el índice de niveles primero, segundo y tercero. El ejemplo 2.4 ilustra el uso de este entorno.

Como se observa, el trabajo manual a realizar es enorme por poco que el índice terminológico crezca. Es por lo que, una vez comprendida la estructura básica del proceso, el comando `\index`, el fichero `idx` y el entorno `theindex`; vamos a exponer con detalle el modo de generar todo ello de forma automática.

2.7.2. El proceso automático con MAKEINDEX

Para generar el índice terminológico con L^AT_EX y MAKEINDEX debemos seguir los cinco pasos que se describen a continuación:

1. En el documento principal:
 - a) incluir el paquete `makeidx` mediante el correspondiente `\usepackage`;
 - b) incluir en el preámbulo del documento el comando⁵

```
\makeindex
```

 - c) incluir, en el lugar donde queramos que aparezca el índice, el comando

```
\printindex
```
2. Incluir en el documento el comando `\index{Entrada!SubEntrada!SubSubEntrada}` con cada *Entrada*, *SubEntrada* y *SubSubEntrada* que queramos introducir en el índice.
3. Compilar el documento. Una vez utilizados los comandos anteriores, la compilación del documento principal produce el fichero de extensión `idx`. Debe tenerse en cuenta que a pesar de que el documento contenga comandos `\index{Entrada}`, el fichero `idx` sólo se escribirá si el comando `\makeindex` se encuentra presente en el preámbulo del documento principal; en otro caso los comandos `\index` utilizados serán ignorados.
4. Generar el índice con formato. Para procesar la información contenida en el fichero `idx` y producir un índice con formato ejecutaremos el programa `MAKEINDEX` (véase el apar-

⁵Los pasos a y b son también necesarios para la generación manual del índice.

tado 2.7.4) pasándole como argumento el fichero `idx`, obtenido en la etapa anterior. Por ejemplo, si nuestro documento es `MiDoc.tex`, ejecutaríamos:

```
makeindex MiDoc.idx
```

Esto produce un nuevo fichero, de extensión `ind`, que contiene ahora la información, pero con comandos `LATEX` y en un entorno `theindex`, todo listo para componer el índice.

5. Volver a compilar el documento principal. Esta segunda compilación hace que el comando `\printindex` utilice el fichero `ind` para imprimir en una nueva página y a dos columnas un impecable índice terminológico ordenado alfabéticamente.

EJEMPLO 2.5

```
\usepackage{makeidx}
\usepackage[spanish]{babel}\makeindex
\begin{document}
[Pág. 2]...Mendeléiev\index{Mendeléiev}...
[Pág. 6]...elemento...sodio\index{sodio},...
[Pág. 10]...las últimas décadas se han
descubierto nuevos metales\index{metales}...
[Pág. 15]...aluminio\index{metales!aluminio}.
[Pág. 18]...Si hablamos del aluminio%
\index{metales!aluminio!densidad}...
\printindex
```

Índices con MAKEINDEX

El ejemplo 2.5 recoge el código necesario para producir el índice del ejemplo 2.4, pero ahora utilizando `MAKEINDEX`.

Las dos formas de generar un índice terminológico producen al final un entorno `theindex`. Éste genera una unidad propia no numerada cuyo título es el valor del comando

```
\indexname
```

El valor que toma este comando en las opciones de `babel` correspondientes a las lenguas del Estado español ha sido ya mostrado en el cuadro 1.2. Puede modificarse con `\renewcommand*` o, si se utiliza `babel`, con `\addto\captionsIdioma`, tal y como se indica en la página 169.

Formato de las entradas La regla general es que las entradas en los índices terminológicos se escriben en minúscula, salvo que se trate de nombres propios; cuando hay varios niveles de entrada, las entradas han de leerse en el orden que determina su nivel. El ejemplo 2.5 ilustra el funcionamiento de la regla.

Esa regla general tiene una excepción: cuando en una entrada con varias palabras que tienen un «orden natural» convenga alterar dicho orden, una coma seguida de una palabra cuya letra inicial sea mayúscula indicará que ahí se debe empezar la lectura, continuando hasta el final de la línea y volviendo al principio de la entrada para terminar (la tabla incluida en este mismo apartado ilustra, con ejemplos, esta situación en su primera columna).

Cuando una entrada tiene una subentrada que debe ser leída en orden inverso, al que fija la regla general para la lectura de entradas y subentradas, la subentrada se escribe con mayúscula

Índice alfabético

Mendeléiev, 2
metales, 10
aluminio, 15
densidad, 18
sodio, 6

inicial, en concordancia con el principio fijado en la excepción antes señalada (véase la segunda columna de la tabla).

Si existe un tercer nivel de entrada, entonces se aplica lo anterior a los dos primeros niveles, para determinar si el segundo de ellos ha de tener mayúscula inicial y, a continuación, los dos primeros se consideran como un grupo único, a efectos de determinar, con el mismo principio, si el tercero debe o no tener mayúscula inicial (véase la tercera columna de la tabla).

enfermedad de Parkinson ... Parkinson, Enfermedad de	página a derecha Estilos de	compilación Modos de sin interrupciones
tamaño del papel ... papel, Tamaño del	peritos López, Juan Roca, Luis	fórmulas Delimitadores en Ajustar tamaños de

2.7.3. Variaciones en la selección de entradas para el índice

Hasta ahora hemos aprendido la sintaxis básica del comando `\index`, con sus posibilidades de selección de entrada a tres niveles. Pero el argumento de este comando admite variaciones con varios objetivos: formas de ordenación alfabética de los términos en el índice, introducción de comandos, rangos de páginas, etc.

Antes de entrar en esta sintaxis extendida de `\index` conviene observar que `\index{Entrada}` debe ir inmediatamente después de la *Entrada* a la que hace referencia, sin espacios entre ambos, y ello para evitar que, al dar formato al documento, pudiera quedar en una página la *Entrada* y en la página siguiente su referencia para el índice, de forma que el número de página que aparecería en el índice como referencia al término *Entrada* sería incorrecto.

A continuación se indica cada uno de los elementos adicionales de la sintaxis de `\index` junto con una breve explicación de los mismos.

<code>\index{Cadena1 see{Cadena2}}</code>	<code>\index{Cadena1 seealso{Cadena2}}</code>
---	---

Ambas sintaxis tienen un efecto similar. Introducen la entrada *Cadena1* en el índice terminológico y, en éste, inmediatamente a continuación de dicha entrada se incluye la frase ‘véase *Cadena2*’ (en el caso de `|see`) o ‘véase también *Cadena2*’ (en el caso de `|seealso`).

Los indicadores `|see` y `|seealso` se expanden en los comandos

<code>\see</code>	<code>\seealso</code>
-------------------	-----------------------

definidos en el paquete `makeidx`. En realidad, para ser más precisos, estos comandos poseen dos argumentos. El resultado que produce la sintaxis `{Cadena1|see{Cadena2}}` en el fichero `ind` es en realidad `\see{Cadena2}{Núm}`, donde `Núm` es el número de página donde se ha introducido dicho `\index`. Cuando se compile el índice, los comandos `\see` y `\seealso` se ejecutarán de acuerdo con su definición, que, por defecto, es:

```
\newcommand*{\see}[2]{\emph{\seename} #1}
\providecommand*{\seealso}[2]{\emph{\aloname} #1}
```

Observe que el segundo argumento, el número de página, es ignorado. Las expresiones ‘véase’ y ‘véase también’ dependen de la definición de los comandos

\seename	\alsoname
----------	-----------

Las citadas expresiones españolas corresponden a la redefinición que la opción `spanish` de `babel` hace de estos dos comandos. Los valores asignados en las otras lenguas del Estado Español se muestran en el cuadro 1.2.

\index{Cadena ()}	\index{Cadena)}
-------------------	------------------

La combinación de estos dos elementos produce la entrada de *Cadena* en el índice, haciendo que la referencia sea, en lugar de a una página aislada, a un intervalo de páginas; la primera del intervalo es la página en la que está `\index{Cadena|()}` y la última aquella en la que se ha incluido `\index{Cadena|)}`. Naturalmente se debe tener cuidado de que la expresión *Cadena* sea exactamente la misma al abrir y al cerrar el intervalo de páginas.

\index{Cadena Comando}

Mediante esta sintaxis se consigue que el número de página correspondiente a la entrada *Cadena* se vea afectado por el comando de nombre `\Comando`. Más concretamente, el resultado que aparece en el fichero IND es `\Comando{Núm}`, siendo *Núm* el número de página. De esta forma podemos conseguir números de página en negrita, itálica, etc., una técnica frecuente para indicar en el índice las entradas principales de un término, por ejemplo. El argumento *Comando* debe ser un nombre de comando, sin la antibarra inicial. Un ejemplo interesante puede ser el siguiente: en [60] se recomienda hacer seguir el número de página en las entradas de un índice terminológico por la abreviatura *n.*, cuando el término de la entrada en cuestión aparece dentro de una nota a pie de página. Esto podría automatizarse con el siguiente código:

```
\newcommand*{\IndPie}[2]{#2\textit{n.}#1}
```

Una vez definido `\IndPie` bastaría utilizar `\index{...|\IndPie{\thefootnote}}` dentro del argumento de `\footnote` donde se encuentra el término para el que se quiere incluir una entrada en el índice.

Si un término del índice aparece en el texto usual de una página y en una nota a pie de la misma página, entonces se recomienda que la entrada contenga el número de página seguido de ‘y *n.*’; para automatizar esto podemos utilizar las referencias cruzadas y el código:

```
\newcommand*{\IndPagPie}[2]{#2\ y\textit{n.}\ref{#1}}
```

Entonces, en el texto, utilizariamos `\index{Cadena|\IndPagPie{EtiqPie}}`, donde *EtiqPie* será la etiqueta asignada, mediante `\footnote{\label{EtiqPie}...}`, a la nota a pie en la que aparece el término *Cadena* y que deseamos reflejar en el índice.

Como se observa, `\see{...}` y `\seealso{...}` son casos particulares de la sintaxis que acabamos de describir.

\index{"Carácter"}

El carácter “ se antepone a cualquier carácter que sea especial para `MAKEINDEX`, concretamente a uno de los tres caracteres !, | y @. De esta forma se consigue que se escriba en el índice cualquiera

de esos caracteres. Si se quiere incluir una palabra que contiene uno o más de estos caracteres, cada uno de ellos debe ser precedido por el símbolo ".

`\index{Cadena1@Cadena2}`

Con el indicador @ conseguimos que se incluya en el índice terminológico el término *Cadena2*, pero que éste se ubique en la posición que correspondería, en el orden alfabético, a *Cadena1*. La utilidad de esta forma de inclusión es enorme. Supongamos que queremos introducir en el índice un término en itálica; lo lógico sería escribir `\index{\textit{Entrada}}` y, efectivamente, funcionará, salvo en algo fundamental: el término *Entrada* no será ordenado alfabéticamente de forma correcta (de hecho, aparecerá en un primer bloque de símbolos). La razón es que MAKEINDEX lee las entradas digamos «literalmente» para ordenarlas alfabéticamente. Sustituyendo el código anterior por `\index{Entrada@\textit{Entrada}}` obtendremos el resultado correcto.

Para saber cuándo es necesario utilizar el indicador @ es conveniente conocer el orden de prelación en la ordenación alfabética que utiliza MAKEINDEX.

Método de alfabetización de MAKEINDEX

Por defecto MAKEINDEX utiliza lo que se llama la ordenación por *palabras*. En la ordenación por *palabras* la prelación es: símbolos, números, blancos, letras mayúsculas y letras minúsculas. Los números se ordenan por su orden natural. Las letras son primero ordenadas independientemente del uso de mayúsculas y minúsculas; cuando dos palabras son idénticas, la que contiene caracteres en mayúscula precede a la que los contiene en minúscula.

El ejemplo 2.6 ilustra las consecuencias de la distinción entre mayúsculas y minúsculas, así como el efecto de los blancos en las *Entradas* de `\index`.

EJEMPLO 2.6

[Pág. 12] \index{sentencia }	Comando, 21
[Pág. 19] \index{comando}	comando, 19
\index{ sentencia}	sentencia, 19
[Pág. 21] \index{Comando}	sentencia , 31
\index{sentencia}	sentencia, 21, 55
[Pág. 31] \index{ sentencia }	sentencia , 12
[Pág. 55] \index{sentencia}	

EJEMPLO 2.7

[Pág. 19] \index{acrata}	Símbolos	M
[Pág. 21] \index{acrata@acrata}	acrata, 19	\makeindex, 42
[Pág. 22] \index{será}	\makeindex, 31	S
[Pág. 23] \index{sera}	A	será, 22
[Pág. 24] \index{sera@será}	acrata, 21	sera, 23
[Pág. 31]		será, 24
\index{\texttt{\textbackslash makeindex}}		
[Pág. 42]		
\index{\makeindex@\texttt{\textbackslash makeindex}}		

MAKEINDEX entiende como símbolo todo carácter que no es un número o una letra del alfabeto inglés. Como consecuencia, los caracteres acentuados, así como algunos habituales en otros idiomas, son considerados como símbolos y no ocupan el lugar que lógicamente podríamos convenir que les corresponde. El ejemplo 2.7 ilustra las dificultades en la ordenación de palabras que comienzan por un símbolo y cómo resolverlas recurriendo al indicador @.

Conviene subrayar aquí que el programa XINDY [42] representa una alternativa a MAKEINDEX para la generación y manejo de índices que ofrece, entre otras, la posibilidad de procesar índices en distintos idiomas, con distintos juegos de caracteres y distintos métodos de alfabetización.

Superposición de indicadores para \index

Los indicadores citados anteriormente, es decir, los caracteres !, " y @, y el carácter que indica el nivel de la entrada (!), pueden utilizarse conjuntamente para realizar *Entradas* de mayor complejidad que las que hemos visto hasta ahora. El el ejemplo 2.8 será más ilustrativo que cualquier descripción verbal que pudiéramos intentar.

EJEMPLO 2.8

[pág. 2] \index{arroba@arroba (\texttt{@})}	ancho, véase también página, 22
\index{comillas@comillas ("")}	arroba (@), 2
[pág. 5] \index{pagina@página (textbf)}	comillas ("'), 2
[pág. 9] \index{pagina@página)textbf}	cigüeña, 22
\index{pagina@página!Estilo de (textit}	página, 5–9, 18
[pág. 12]	Ancho de la, 12
\index{pagina@página!ancho de@Ancho de la textit}	Estilo de, 9–18
[pág. 18] \index{pagina@página}	
\index{pagina@página!Estilo de)textit}	
[pág. 22] \index{ancho textbf}	
\index{ancho seealso{página}}	
\index{cigüeña@cig\"ueña}	

Indicadores combinados en el comando \index

Si analiza este ejemplo comprobará que el término ‘ancho’ aparece en dos \index, situados en la misma página. En realidad esto es incorrecto, y MAKEINDEX emitirá un mensaje de precaución o Warning (los mensajes de precaución y errores detectados por MAKEINDEX se emiten en pantalla y se graban en el fichero de extensión *i1g*). Concretamente se indica que hay un conflicto debido a la multiplicidad de indicadores para la misma entrada en la misma página (entrada ‘ancho’ en la página 22). Aunque el resultado es aceptable, este mensaje es molesto, y, además, parece razonable que el número de página preceda a la llamada ‘véase también’. Todo ello se debe a que no es posible simultanear los indicadores |see o |seealso con el indicador |Comando. De hecho, como ya hemos visto, |see o |seealso ignoran el número de página, aunque esto puede ser corregido redefiniendo \see o \seealso, por ejemplo, en la forma siguiente:

```
\renewcommand*{\seealso}[2]{\#2,\ \emph{\alsoname}\ #1}
```

Todavía podríamos mejorar la situación, para posibilitar la inclusión de comandos que afecten al número de página, mediante un nuevo comando; por ejemplo, con

\newcommand*{\MiAlso}[3]{#2{#3}, \ \emph{\alsoname}\ #1} podríamos escribir \index{Entrada|MiAlso{Término}{Comando}}; por ejemplo, para el caso comentado escribiríamos \index{ancho|MiAlso{página}{\textbf{}}}.

Precauciones

Una *Entrada* de cualquier nivel, es decir, el argumento de \index, puede contener comandos, como ya hemos comprobado. Sin embargo conviene tener en cuenta algunas observaciones:

- los comandos incluidos se ejecutarán en el momento de imprimir el entorno `theindex`, y no al leer el comando \index;
- la regla anterior no se aplica cuando \index aparece en el argumento de otro comando; en este caso el argumento de \index será ejecutado y es el resultado de dicha ejecución el que será ordenado alfabéticamente por `MAKEINDEX`;
- el código \index{\{} produce un error, pues L^AT_EX busca, sin encontrar, al compañero \};
- el comando \verb puede aparecer en el argumento de \index, un caso excepcional (véase el apartado PARA SABER MÁS del capítulo 1), salvo que \index esté, a su vez, en el argumento de otro comando;
- si los comandos incluidos son frágiles, éstos deben ser protegidos con el comando \protect (véase la sección 2.2.2).

Observe que una situación típica en la que \index aparece dentro de un comando es la de una nota a pie de página. La sintaxis \index{...|IndPie{\thefootnote}}, presentada en la página 236, funciona porque aprovecha que \thefootnote es ejecutado en el momento de leer \index, ya que éste forma parte del argumento de un comando \footnote.

EJEMPLO 2.9

```
\newcommand*{\Escribe}[1]{#1}
[pág. 4]
\Escribe{Palabra}\index{\Escribe{Palabra}}
    \index{"@}
[pág. 6]\Escribe{Palabra}
    \index{\Escribe{Palabra}}
```

Fichero ind	Índice alfabético
\begin{theindex}	
\item @, 4	@, 4
\item \Escribe{Palabra}, 6	Palabra, 6
\indexspace	
\item Palabra, 4	Palabra, 4
\end{theindex}	

Múltiples entradas de un término

El paquete varindex, creado por Martin Väth [67], proporciona una potentísima herramienta para generar entradas complicadas en el índice terminológico. Desgraciadamente los problemas de espacio nos impiden presentarlo aquí, incluso someramente. Nos contentaremos con dar una idea de lo que es capaz. Supongamos que deseamos introducir dos entradas: ‘título’, con subentrada ‘libro’, y ‘libro’, con subentrada ‘título’. En la forma estándar necesitamos dos comandos \index:

```
\index{titulo@título!libro}
\index{libro!titulo@título}
```

mientras que con varindex un único comando es suficiente. Esto no sería casi nada si fuera todo. Quizás el siguiente ejemplo, basado en otro de [67], pueda ser más ilustrativo.

EJEMPLO 2.10

```
Observe el significado de los números en:  
\Index{medida de no compacidad de Hausdorff}[%  
[34,12 1!~234!6= ]  
34,12: tercera y cuarta palabras,  
        primera y segunda  
1!~234: primera palabra  
        subentrada: $\sim$  
        segunda tercera y cuarta  
!6=: subsubentrada: sexta $\approx$  
(espacio final): todas las palabras
```

medida	
~ de no compacidad	
Hausdorff ≈, 17	
medida de no compacidad de Hausdorff, 17	
no compacidad, medida de, 17	

2.7.4. MAKEINDEX

MAKEINDEX es un programa de propósito general para generar índices jerárquicos, diseñado y escrito en lenguaje C por Pehong Chen [13]. En esta sección describimos las principales opciones en el momento de ejecutar MAKEINDEX. La sintaxis general de la línea de comandos para ejecutar el programa es:

```
MAKEINDEX [-c] [-l] [-o Nom] [-p Num] [-r] [-s Sty] [-t Log] MiDoc.idx
```

donde *MiDoc.idx* es el nombre del fichero *idx* que le pasamos como argumento a MAKEINDEX. Podemos pasar simultáneamente varios ficheros *idx* a MAKEINDEX. Las opciones indicadas de la línea de comandos tienen el efecto descrito a continuación.

- c** Comprime los espacios en blanco que ocupan posiciones intermedias y suprime los situados al principio y final de la *Entrada*.
- l** Provoca que MAKEINDEX utilice el orden por *letras*. En este orden, a diferencia del orden por *palabras* utilizado por defecto, los blancos no se tienen en cuenta. En la ordenación por palabras, el término ‘para psicólogos’ precede a ‘parapsicólogos’, mientras que en la ordenación por letras el orden se invierte.
- o Nom** La salida se escribe en el fichero *Nom*. La cadena *Nom* puede contener un camino válido. Esta salida contiene el índice con los comandos de formato (*theindex*, *\item*, etc.); es el fichero *ind* en la ejecución estándar.
- p Núm** Provoca que el número de página asignado a la primera página del índice sea el indicado por *Núm*, lo que puede ser útil cuando el índice se va a componer por separado.
- r** Inhabilita el método implícito para la formación de intervalos de páginas. Por defecto, tres o más páginas sucesivas son abreviadas automáticamente en forma de intervalo (1–6). Cuando se ejecuta MAKEINDEX con esta opción *-r* la única forma de producir intervalos de páginas es declarándolos explícitamente con |(...)|.
- s Sty.ist** MAKEINDEX puede basar su comportamiento en un fichero de estilo. Esta opción indica que se utilice el archivo de estilo de nombre *Sty*, y extensión *ist*, que es la extensión

de este tipo de archivos. *Sty* puede contener el camino para localizarlo. Es posible omitir el camino en *Sty*, indicando sólo el nombre del fichero y la extensión, si éste está ubicado donde indica la variable de entorno INDEXSTYLE, que debe estar, en este caso, cargada en nuestro sistema.

-t Log Emplea *Log* como fichero en el que escribir los errores e información general de la ejecución. Por defecto, el nombre del fichero se crea tomando el nombre *MiDoc.idx* y sustituyendo la extensión *idx* por la extensión *ilg*.

2.7.5. Estilos para MAKEINDEX

Como se ha indicado en el apartado anterior, MAKEINDEX puede utilizar un fichero de estilo para adecuar la presentación del índice a nuestro gusto o necesidades. En el apartado siguiente veremos otra forma de influir en la presentación del índice consistente en la modificación del propio entorno *theindex*.

A continuación presentamos una breve descripción del contenido de los ficheros de estilo para MAKEINDEX. En general estos ficheros de estilo, normalmente con extensión *ist*, contienen una serie de palabras clave a las que se les asignan ciertos valores. Las palabras clave se pueden clasificar en dos grupos: las utilizadas para operaciones de *entrada*, o lectura, y las utilizadas para operaciones de *salida*, o escritura. Entre las primeras podemos encontrar, por ejemplo, la que especifica el carácter utilizado para indicar el nivel en el índice (!), y entre las segundas, encontraremos algunas destinadas a la presentación final del índice. Si deseamos escribir nuestros propios ficheros de estilo conviene saber que las palabras clave pueden colocarse en un orden arbitrario y que aquellas que no especifiquemos en el archivo tomarán su valor por defecto.

En la lista que figura a continuación cada palabra clave se presenta seguida de su valor por defecto, que aparece sombreado. En general, el valor asignado a cada palabra clave debe ser encerrado entre delimitadores: si es un único carácter se delimita por comillas simples de cerrar, es decir, la sintaxis sería '*Car*', donde *Car* es cualquier carácter; si es una cadena de caracteres se encierra entre dobles comillas, es decir, "*Cadena*". Cada palabra clave espera un valor que puede ser de tipo carácter, de tipo cadena o numérico (entero positivo), lo que se indica mediante [c], [s] o [n]. Si la cadena que se asigna como valor debe contener alguno de los caracteres especiales \, ' o ", cada uno de ellos debe ser precedido por \, lo que, en particular, explica la profusión de la cadena \\\. Al igual que en LATEX, el carácter % indica que lo que le sigue, hasta el final de la línea, es un comentario y debe ser ignorado.

Parámetros de entrada

Las siguientes palabras clave determinan el contenido del fichero *idx*.

keyword [s] "\\indexentry" Informa a MAKEINDEX que su argumento es una entrada para el índice.

arg_open [c] '{' Delimitador inicial de un argumento.

arg_close [c] '}' Delimitador final de un argumento.

range_open [c] '(' Delimitador inicial del rango de páginas.

`range_close [c] ')` Delimitador final del rango de páginas.

`level [c] ?!` Indicador de nuevo nivel o subentrada.

`actual [c] @` Carácter que indica que el resto del argumento será el que se imprima en la entrada del índice.

`encap [c] |` Carácter que indica que el resto del argumento va a ser expandido como un comando que actuará sobre el número de página de la entrada (denominado «encapsulador»).

`quote [c] "` Carácter que toma el carácter siguiente literalmente.

`page_compositor [s] " -` Separador de números de página compuestos. Si el contador de las páginas a las que se debe hacer referencia en el índice se representa mediante números compuestos de dos números, este carácter es el que `MAKEINDEX` reconoce como separador de ambas numeraciones.

Parámetros de salida

Las siguientes palabras clave determinan el contenido del fichero `ind`. Se encuentran agrupadas según su función. Los términos `\n` y `\t` son comandos internos que consisten en una línea en blanco y una sangría, respectivamente.

- *Generales*

`preamble [s] "\\begin{theindex}\n"` Comienzo del fichero `ind`, precede a todas las entradas (`\item`, `\subitem` y `\subsubitem`). Su valor puede incluir todos los comandos necesarios para dar un formato al índice compatible con el resto del documento, lo que puede ser muy útil cuando el índice va a ser compilado independientemente del documento.

`postamble [s] "\n\n\\end{theindex}\n"` Final del fichero `ind`. Su valor se incluye después de todas las entradas.

- *Primera página*

`setpage_prefix [s] "\n \\setcounter{page}{}` Prefijo para el número de la primera página (proporcionado por el parámetro de ejecución `-p`).

`setpage_suffix [s] "}\n"` Sufijo para el número de la primera página.

Inicio de grupo alfabético.

`group_skip [s] "\n\n \\indexspace\n"` Cadena que se inserta antes de un nuevo grupo de letra; usualmente se expandirá en un comando que inserte un espacio vertical.

`heading_prefix [s] ""` Prefijo para la cabecera de un nuevo grupo de letra.

`heading_suffix [s] ""` Sufijo para la cabecera de un nuevo grupo de letra.

`headings_flag [n] 0` Interruptor que determina si cada grupo de letra posee una cabecera. Si su valor es 0 no se inserta nada. Se coloca una cabecera en mayúsculas o minúsculas según tenga valor positivo o negativo. En ambos casos el valor de la cabecera va precedido por `heading_prefix` y seguido por `heading_suffix`.

`symhead_positive [s] "Symbols"` Cabecera para el grupo de símbolos que será insertada si `headings_flag` es positivo.

`symhead_negative [s] "symbols"` Cabecera para el grupo de símbolos que será insertada si `headings_flag` es negativo.

`numhead_positive [s] "Numbers"` Cabecera para el grupo de números que será insertada si `headings_flag` es positivo.

`numhead_negative [s] "numbers"` Cabecera para el grupo de números que será insertada si `headings_flag` es negativo.

- *Separadores de entradas*

`item_0 [s] "\n \\item "` Comando que se insertará entre dos entradas de nivel 0.

`item_1 [s] "\n \\subitem "` Comando que se insertará entre dos entradas de nivel 1.

`item_2 [s] "\n \\subsubitem "` Comando que se insertará entre dos entradas de nivel 2.

`item_01 [s] "\n \\subitem "` Comando que se insertará entre una entrada de nivel 0 y una de nivel 1.

`item_12 [s] "\n \\subsubitem "` Comando que se insertará entre una entrada de nivel 1 y una de nivel 2.

`item_x1 [s] "\n \\subitem "` Comando que se insertará entre una entrada de nivel 0 y una de nivel 1 cuando la de nivel 0 no tiene número de página.

`item_x2 [s] "\n \\subsubitem "` Comando que se insertará entre una entrada de nivel 1 y una de nivel 2 cuando la de nivel 1 no tiene número de página.

- *Delimitadores de números de página*

`delim_0 [s] ", "` Separador entre la entrada y el número de página, a nivel 0.

`delim_1 [s] ", "` Separador entre la entrada y el número de página, a nivel 1.

`delim_2 [s] ", "` Separador entre la entrada y el número de página, a nivel 2.

`delim_n [s] ", "` Separador entre dos números de página a cualquier nivel.

`delim_r [s] "--"` Separador entre dos números de página que indican un intervalo.

`suffix_2p [s] ""` Cadena que sustituirá al separador de intervalo y al número de la segunda página en una lista de dos páginas consecutivas. Si se incluye reemplaza a `delim_r`.

`suffix_3p [s] ""` Cadena que sustituirá al separador de intervalo y al número de la última página en una lista de tres páginas consecutivas. Si se incluye reemplaza a `delim_r` y a `suffix_mp`.

Un valor típico, en español, sería " y ss ." para indicar sucesivas páginas.

`suffix_mp [s] ""` Cadena que sustituirá al separador de intervalo y al número de la última página en una lista de tres o más páginas. Si se utiliza reemplaza a `delim_r`.

`delim_t [s] ""` Delimitador que se imprimirá al final de la lista de páginas de cada entrada.

- *Encapsulados*

`encap_prefix [s] "\\"` Prefijo de la palabra que sigue al indicador | de \index («encapsulado»).

`encap_infix [s] "{"` Prefijo del número de página cuando se utiliza un encapsulado.

`encap_suffix [s] "}"` Sufijo del número de página cuando se utiliza un encapsulado.

- *Ordenación de páginas*

`page_precedence [s] "rRnaA"` Ordenación de la referencia a las páginas, según la representación de su contador: a y A denotan representación alfabética en minúsculas y mayúsculas, n numérico, r y R minúsculas y mayúsculas romanas.

El código siguiente, autoexplicativo, podría ser el contenido de un fichero de estilo, con cambios tanto en las palabras clave de entrada como de salida. Observe que se cambia el indicador de

nivel de la entrada estándar (!) por el carácter >. El ejemplo 2.11 muestra el resultado de este estilo sobre un índice con algunos términos de esta sección (ejecutando MAKEINDEX con parámetro -s MiEstilo.ist).

```
%%.Fichero 'MiEstilo.ist'
level '>' %Sustituye al separador de nivel estándar !
preamble "\n \begin{theindex} \n"
postamble "\n\n \end{theindex}\n"
delim_0 "\\dotfill "
delim_1 "\\dotfill "
item_1 "\\item --- "
item_01 "\\item --- "
item_x1 "\\item --- "
heading_prefix "\\centerline{\\bfseries " %Línea centrada que inicia grupo
heading_suffix "}\\\nopagebreak\n" %Termina el argumento de la línea
headings_flag 1 %Activa el preámbulo de cada grupo, en mayúscula
symhead_positive "S'\\'\\i mbolos'" %Cabecera símbolos
suffix_mp " y ss." %Sufijo para intervalos de tres o más páginas
% Final de fichero 'MiEstilo.ist'
```

EJEMPLO 2.11

```
\index{indicetermin@índice terminológico}...
\index{indicetermin@índice terminológico>formato}...
...
\index{makeindex@{\textsc{makeindex}}, Programa|{}}
... \index{makeindex@\verb+\makeindex+}
... \index{index@\verb+\index+}
... \index{index@\verb+\index+>Blancos en}...
\index{index@\verb+\index+>indicadores}...
\index{"|}@{\texttt{|}} (pleca)...
\index{makeindex@{\textsc{makeindex}}, Programa|{})}
```

Símbolos

(pleca)	235
I	
\index	232
— Blancos en	237
— indicadores	235
índice terminológico	231
— formato	233
M	
MAKEINDEX, Programa	231 y ss.
— \makeindex	233

Cambios en el estilo del índice terminológico; los números de página corresponden a los de este libro

En el código del ejemplo 2.11 puede resultarle incomprendible el modo en el que aparece escrito el término «Símbolos». El origen de esta complicación reside en la forma con la que T_EX maneja internamente los caracteres acentuados para garantizar la compatibilidad entre plataformas (consulte el apartado PARA SABER MÁS del capítulo 1).

2.7.6. Pequeñas modificaciones en el entorno theindex

En el apartado anterior hemos comentado cómo es posible utilizar un fichero de estilo de MAKEINDEX para poder escribir ciertos comandos en el fichero ind, antes y después de iniciado el entorno theindex, y así poder modificar el comportamiento de este entorno. Mediante el comando

\renewenvironment podemos modificar directamente el entorno `theindex`. Esta estrategia nos permite realizar cambios en la composición del índice terminológico que no son posibles con los cambios en el fichero de estilo de `MAKEINDEX`. Naturalmente las posibilidades son enormes, por lo que nos limitaremos a dos ejemplos sencillos.

Si estamos utilizando, por ejemplo, la clase `article` y al componer un documento queremos incluir un índice terminológico, parece lógico que éste no inicie una nueva página y que sea tratado como una sección dentro del artículo. Esto lo podemos conseguir evitando la inclusión del comando `\twocolumn`, que la definición por defecto incluye, y sustituyéndolo por un entorno `multicols`, con el número de columnas adecuado (que usualmente será dos).

```
\makeatletter
\renewenvironment{theindex}{%
  \section*{\indexname}%
  \noindent En este índice ... %(possible introducción al índice)
  \addcontentsline{toc}{section}{\indexname}%
  \let\item\@idxitem%
  \begin{multicols}{2}%
  \end{multicols}}%
\makeatother
```

Algunas observaciones sobre el código anterior son pertinentes. En primer lugar observe que el entorno `theindex` inicia una sección con asterisco, asignándole el título guardado en `\indexname`. Puesto que dicha sección (`\section*`) no incluye la correspondiente entrada en el índice general, recurrimos explícitamente a un comando `\addcontentsline`. La materialización del formato a dos columnas se consigue con el entorno `multicols`, que no inicia nueva página (véase la lección 12). Finalmente, observe la línea en la que se define `\item` como `\@idxitem`; es una asignación interna necesaria, de carácter técnico, que nos obliga a encerrar parte del código entre `\makatletter` y `\makeatother` (para el significado de `\let` véase la sección 8.1.5).

En un documento de la clase `book` procederíamos de forma similar, sustituyendo `\section*` por `\chapter*`. Como consecuencia, el índice terminológico se iniciaría en la siguiente página a derecha (a no ser que se hubiera incluido la opción `openany` de la clase).

Finalmente puede ser interesante conocer las definiciones de algunos comandos internos que hemos manejado en toda esta sección. Las siguientes definiciones se encuentran (en una forma equivalente) incluidas en las clases de documento estándar:

```
\newcommand*\@idxitem{\par\hangindent 40pt}
\newcommand*\subitem{\@idxitem \hspace*{20pt}}
\newcommand*\subsubitem{\@idxitem \hspace*{30pt}}
\newcommand*\indexspace{\par
  \vskip 10pt plus 5pt minus 3pt\relax}
```

El significado de `\hangindent` se encuentra en la sección 8.4 y en cuanto al comando

`\indexspace`

está básicamente destinado, como muestra su definición, a separar entre sí los grupos alfabéticos.

2.7.7. Varios índices terminológicos

El paquete `index`, desarrollado por David Jones [30], lleva a cabo una doble tarea: por un lado, redefine los comandos adecuados de `LATEX` para proporcionar al usuario una mejor gestión de los índices y, por otro, aporta herramientas sencillas para la construcción de dos o más índices terminológicos en un mismo documento. Esta última posibilidad es, naturalmente, de máximo interés, especialmente en algunas disciplinas; pensemos, por ejemplo, en los índices topográficos, onomásticos, cronológicos, etc. Este libro, sin ir más lejos, dispone de dos índices terminológicos, como usted ya ha podido comprobar, además, se generaron con el paquete `index`.

El paquete utiliza un procedimiento diferente del estándar de `LATEX` para generar los índices. La diferencia es esencial, y en ella reside la gran potencia de este paquete. Para generar el índice, `index` recurre al fichero auxiliar `aux`, donde se guardan las entradas que deben ir al índice (más precisamente, a cualquiera de los índices) y desde el que se copian al fichero `idx` (o al correspondiente `adx`, véase el comando `\newindex` un poco más adelante). De esta forma, y esto es de un valor incalculable, si estamos compilando un documento largo consistente en varios ficheros incorporados con comandos `\include`, los índices se mantendrán actualizados e incrementados conforme vayamos compilando los distintos ficheros incluidos y no perderemos el índice si sólo compilamos alguno de ellos con el correspondiente comando `\includeonly`.

Los nuevos índices terminológicos son creados por el primero de los comandos que figuran a continuación, mientras que el segundo de ellos permite redefinir un índice ya existente.

```
\newindex{NombreÍndice}{adx}{and}{TítuloÍndice}
\renewindex{NombreÍndice}{adx}{and}{TítuloÍndice}
```

El comando `\newindex` sólo puede incluirse en el preámbulo del documento. El significado de los cuatro argumentos es el que figura a continuación.

NombreÍndice es una etiqueta identificativa, que el usuario asigna al índice que desea definir y que permite distinguirlo de los otros que construya en el mismo documento. Esta etiqueta no tiene nada que ver con el nombre del fichero en el que se guardarán las entradas para el índice, ni con el antetítulo del índice impreso.

adx se utiliza como extensión del primer fichero generado para contener los términos del índice *NombreÍndice*. Si compilamos el fichero fuente `MiDoc.tex`, tendremos un fichero `MiDoc.adx`, en el que se habrán almacenado las entradas al índice *NombreÍndice*. Es el análogo al fichero `idx` en el caso de los índices estándar.

and se utiliza como extensión del fichero que contiene las entradas del índice *NombreÍndice* con formato. El proceso para generar este índice con formato consiste ahora en ejecutar⁶:

```
makeindex -o MiDoc.and MiDoc.adx
```

⁶Si usted trabaja con un sistema integrado para `LATEX`, como alguno de los comentados en la lección 2, debe prestar atención. La ejecución de `MAKEINDEX` con ayuda de un botón o comando de menú de su sistema puede que se realice con algunos parámetros fijados, y en este caso quizás necesite configurar el sistema para que admita, por ejemplo, otros ficheros de entrada y salida.

Con ello, `MAKEINDEX` tomará las entradas de `MiDoc.adx` y las escribirá, junto con comandos de `LATEX`, en `MiDoc.and`.

TítuloÍndice es el título que se dará a la unidad del índice *NOMBREÍNDICE* cuando se imprima.

Para poder enviar entradas a los nuevos índices el paquete redefine los comandos `\index` y `\printindex`, la nueva sintaxis es la siguiente:

<code>\index[NOMBREÍNDICE]{Entrada}</code> <code>\index*[NOMBREÍNDICE]{Entrada}</code>	<code>\printindex[NOMBREÍNDICE]</code>
---	--

El argumento *NOMBREÍNDICE* es la etiqueta identificativa (del comando `\newindex`) del índice al que se enviará la *Entrada*. Este último argumento tiene el mismo significado que en el original `\index`, admitiendo, naturalmente, la posibilidad de los tres niveles de entrada y el uso de indicadores. El comando `\printindex` necesita también la etiqueta del índice que debe imprimir.

La nueva versión con asterisco de `\index`, `\index*`, además de enviar su argumento *Entrada* al índice *NOMBREÍNDICE*, escribe dicho argumento en el lugar donde aparece el comando. El argumento *Entrada* se escribe sin modificación alguna; es decir, los posibles caracteres !, @... serán impresos y los comandos que pueda contener dicho argumento serán ejecutados.

EJEMPLO 2.12

```
\usepackage[spanish]{babel}
\usepackage{makeidx}\usepackage{index}
\newindex{ono}{odx}{ond}{Índice onomástico}
\newindex{topo}{tdx}{tnd}{Índice toponímico}
\newcommand*\{Lugar}[2]{\#1\index[topo]{\#2@#1}}
\renewcommand*\{seealso}[2]%
{#2, \emph{\alsoname} #1}
\makeindex\begin{document}...
\index[ono]{Abencerrajes\seealso{Ban\=u Sarr\=a\^y}}
\Lugar{Adén}{aden}...
\Lugar{Alhama de Granada}{alhamadegranada}
\Lugar{Alhama de Murcia}{alhamademurcia}...
\Lugar{Alhama de Granada}{alhamadegranada}...
\index[ono]{abbad@\`Abb\=ad (liberto)}...
\index[ono]{abbadia@al-\`Abb\=add\=i, A.M.}...
\Lugar{Afganistán}{afganistan}...
\index[ono]{abbasihsan@\`Abb\=as, I\d{h}s\=an}...
\printindex[ono]\printindex[topo]
```

Índice onomástico

'Abbād (liberto), 360
 al-'Abbādī, A. M., 373
 'Abbās, Ihśān, 410
 Abencerrajes, 41, véase también Banū Sarrāŷ
 ...

Índice toponímico

Adén, 251
 Afganistán, 385
 Alhama de Granada, 305, 314
 Alhama de Murcia, 305

Índices onomástico y toponímico con `index` (extracto de los índices de *Historia de España*, dirigida por Manuel Tuñón de Lara, Vol. 3, *España musulmana*, Ed. Labor, 7.^a reimpresión, 1988). Para comprender el código de algunos de los caracteres acentuados consulte el apartado PARA SABER MÁS del capítulo 1

El paquete `index` tiene un pequeño error sobre el que conviene estar prevenido. La ejecución de `\index` se realiza mediante una técnica especial, consecuencia de la cual es que cada comando presente en el argumento *Entrada* de `\index` se incluye en el fichero aux (y luego en `adx` y `and`) seguido de un espacio. En general este espacio es totalmente inocente, pero con algunos comandos especiales, por ejemplo con `\verb`, produce salidas no deseadas.

¿Por qué *NOMBREÍNDICE* es un argumento optativo de estos comandos? ¿No parece más natural que fuera obligatorio? La razón es que la sintaxis anterior no cambia la estándar del `\index`

de L^AT_EX, sino que la extiende. Si no se incluye este argumento optativo todo ocurre como si estuviéramos manejando un único índice, idéntico al estándar que hemos venido describiendo en esta sección. De hecho, este índice estándar es «directamente accesible», mediante una etiqueta identificativa especial, que define el paquete. Esta etiqueta es `default`; más aún, el comando `\makeindex` se redefine en el paquete `index` como:

```
\newindex{default}{idx}{ind}{\indexname}
```

Una vez guardado el código del ejemplo en el documento `MiDoc.tex` y compilado, para obtener el índice con formato sería necesario ejecutar dos veces `MAKEINDEX` en la forma:

```
\makeindex -o MiDoc.ond MiDoc.oxd
\makeindex -o MiDoc.tnd MiDoc.tdx
```

y al compilar el documento otra vez obtendríamos el resultado del ejemplo 2.12.

A continuación se describen otros comandos que incorpora el paquete `index` para facilitar la creación y manipulación de los índices.

<code>\shortindexingon</code>	<code>\shortindexingoff</code>
-------------------------------	--------------------------------

Estas declaraciones activan y desactivan, respectivamente, los caracteres `^` y `_` como abreviaciones para `\index*` e `\index`. Las declaraciones tienen efecto local, de forma que pueden ser usados para habilitar o inhabilitar dichas abreviaciones. Por razones evidentes estas abreviaciones no están disponibles en modo matemático. También existe el entorno `shortindexingon`.

<code>\proofmodetrue</code>	<code>\proofmodefalse</code>	<code>\indexproofstyle{Opciones}</code>
-----------------------------	------------------------------	---

Permiten incorporar y controlar una funcionalidad especial: la escritura de *Etiqueta*, para cada comando `\index`, como nota al margen⁷. Esta utilidad es una ayuda en la gestión del índice, fundamentalmente para alcanzar la coherencia de los términos que deseamos que aparezcan en el mismo. Estas *Entradas* al margen se escribirán utilizando la fuente `typewriter` en tamaño `\footnotesize`. El registro `\indexproofstyle` puede ser utilizado para cambiar el tipo y tamaño de fuente, por ejemplo mediante

```
\indexproofstyle{\tiny\ttfamily}
```

<code>\disableindex{Lista}</code>

se utiliza para evitar la escritura del fichero `adx` donde se guardan las entradas al índice. El argumento *Lista* debe ser una lista de las etiquetas *NOMBREÍNDICE* de los índices que no se quieren escribir, separadas por comas. Debe ser anterior a las declaraciones `\newindex` de los índices.

2.7.8. Glosarios con MAKEINDEX

El término *glosario*, según la primera acepción del *Diccionario de la Lengua de la Real Academia Española* significa: «catálogo de palabras oscuras o desusadas, con definición o explicación de cada una de ellas». En cambio, en [60], se extiende el concepto a «voz dudosas, técnicas, os-

⁷Una funcionalidad similar se obtiene con el paquete `showidx`, que no describiremos aquí, véase [5] o [11].

curas, dialectales, jergales, etc.», pero no se menciona la necesidad de una «explicación», algo que, en nuestra opinión, es imprescindible, a no ser que tratemos el glosario como otro índice terminológico.

\LaTeX proporciona ayudas para la generación de un glosario, muy similares, aunque con algunas ausencias importantes, a las proporcionadas para el índice alfabético. He aquí una breve descripción de las mismas. Los comandos básicos son los siguientes:

<code>\makeglossary</code>	<code>\glossary{Entrada!SubEntrada!SubSubEntrada}</code>
----------------------------	--

El primero de ellos, el análogo a `\makeindex`, crea un fichero de nombre el del documento compilado y extensión `glo` (que cumple el mismo papel que el fichero `idx` para el índice alfabético). El segundo comando, como se ve, es el análogo de `\index`. Cada comando `\glossary` genera una entrada en el fichero `glo` del tipo

$$\text{\glossaryentry}{Entrada!SubEntrada!SubSubEntrada}{N\'um}$$

donde `N\'um` es el número de la página donde se encuentra `\glossary` (es decir, `\glossaryentry` es el sustituto de `\indexentry`).

Ahora bien, aquí se acaba la ayuda de \LaTeX para los glosarios. Para concluir el trabajo y poder recurrir a `MAKEINDEX` a fin de obtener automáticamente un glosario adecuado, debemos realizar los siguientes pasos:

1. Definir un fichero de estilo para `MAKEINDEX`, que podríamos llamar `glosario.ist`, y que debe contener al menos las siguientes líneas:

```
keyword "\glossaryentry"
preamble "\n \begin{theglossary}\n"
postamble "\n\n \end{theglossary}\n"
```

2. Definir en el preámbulo del documento un comando encargado de imprimir el glosario; podemos darle el nombre que deseemos, digamos `\printglossary`:

```
\makeatletter
\newcommand*\printglossary{\@input{\jobname.gld}}
\makeatother
```

donde la extensión `gld` puede ser, naturalmente, sustituida por cualquier otra (la definición anterior es análoga a la de `\printindex` en el paquete `makeidx`).

3. Definir el entorno `theglossary`. Las posibilidades son múltiples, pero sin duda convenaremos en que un entorno `description`, o uno personalizado de tipo `list`, es adecuado. Además, este entorno debería comenzar una unidad, del nivel adecuado a la clase de documento usada, que colocara el título guardado en el comando

<code>\glossaryname</code>

cuya definición en las lenguas del Estado español se encuentra en el cuadro 1.2, página 171.

4. Realizados los pasos anteriores, ejecutaríamos `MAKEINDEX` en la forma

$$\text{makeindex -s Glosario.ist -o MiDoc.gld MiDoc.glo}$$

Probablemente será también conveniente redefinir el comando `\@idxitem` para dar formato adecuado a las entradas del glosario (que pueden ser largas si se incluye la explicación de cada término, lo que no ocurre en los índices). O, quizás, podríamos incluir cada término en la forma

```
\glossary{EntradaOrd@\textbf{Entrada}: }Explicación}
```

obteniendo, con un entorno de tipo lista para `the glossary`, un resultado adecuado.

En la sección 2.8.5 puede encontrar otra forma de abordar los glosarios. La técnica allí empleada es diferente: no hace uso de `MAKEINDEX`, está basada en la utilización de recursos diseñados para la gestión automatizada de listas bibliográficas. Es una alternativa que debe tenerse muy en consideración si se está interesado en los glosarios.

2.8. Mecanizar la bibliografía

En la lección 15 se describió el comando `\cite` para referirse a los ítems de una lista bibliográfica. Dicha lista se construye con el entorno `the bibliography` y debe contener un ítem para cada uno de los comandos `\cite` del documento, identificado mediante la etiqueta correspondiente, pudiendo contener además otros ítems. Al autor corresponde el trabajo de ordenar y dar un formato homogéneo a los diferentes ítems que figuran en la lista, atendiendo a costumbres de uso que dependen de las disciplinas. En esta sección describimos nuevas herramientas para que sea el computador, mediante el programa `BIBTEX`, el encargado construir y mantener el entorno `the bibliography`, de acuerdo con los comandos `\cite` que el autor introduzca en el documento. Para ello se utilizarán dos elementos, una *base de datos* de bibliografía, que contendrá en general muchas más «fichas» de las que serán usadas en el documento y que podrá ser utilizada en otros documentos, y un *estilo de bibliografía*, seleccionado de entre un conjunto, que conoce y aplica esas costumbres de uso a las que nos hemos referido. El autor se limitará a ir añadiendo nuevas «fichas» o registros en la base de datos cada vez que necesite un nuevo elemento bibliográfico que no se hubiera incluido con anterioridad.

El entorno `the bibliography`, que es, con independencia de cuál haya sido su génesis, el utilizado por `LATEX` para la lista bibliográfica, está construido internamente como una unidad de estructura con asterisco (`chapter` en la clase `book` y `section` en la clase `article`) y tiene un título. Dicho título es el valor asignado a los comandos

<code>\bibname</code>	<code>\refname</code>
-----------------------	-----------------------

El primero se utiliza en la clase `book` y el segundo en `article`. El valor que toma este comando en las opciones de `babel` correspondientes a las lenguas del Estado español ha sido ya mostrado en el cuadro 1.2. Puede ser modificado con `\renewcommand*`, o, si se utiliza `babel`, con `\addto\captionsIdioma`, tal y como se indica en la página 169.

En lo que a las marcas se refiere, `LATEX` utiliza el comando `\MakeUppercase` para poner los títulos en mayúscula; si este comportamiento no es de su agrado, en las secciones 2.4 y 2.4.1 encontrará la información necesaria para cambiarlo.

Pero `LATEX`, sorprendentemente, no incluye en el índice general la entrada correspondiente a la bibliografía, por lo que habrá de ser el usuario quien lo haga. Una forma de hacerlo consiste

en redefinir en el preámbulo de nuestro documento el entorno `thebibliography`. Las líneas que siguen tienen es propósito.

```
\let\OLDthebibliography=\thebibliography
\def\thebibliography#1{\OLDthebibliography{#1}
  \addcontentsline{toc}{chapter}{\bibname}}
```

El comando `\let` (véase la sección 8.1.5) se utiliza en el código anterior para crear un nuevo comando, `\OLDthebibliography`, en el que almacenamos la definición inicial del comando `\thebibliography`. A continuación, mediante el comando `\def` (véase la sección 8.1), definimos de nuevo el comando `\thebibliography`, de tal manera que a la definición inicial, cualquiera que fuera, le añadimos un comando `\addcontentsline` (véase la sección 2.2.1) para incluir la línea deseada en el índice general. El código para la clase `article` únicamente difiere del anterior en que se sustituye `chapter` por `section` y `\bibname` por `\refname`.

2.8.1. BIBTEX

`BIBTEX` es un programa diseñado por Oren Patashnik [53, 54] para generar de forma automática, en combinación con `LATEX`, el entorno `thebibliography` a partir de los comandos `\cite` de nuestro documento y de bases de datos y estilos de bibliografía creados para `BIBTEX`. De esta manera se logra:

Consistencia en la presentación de la lista de referencias. Por ejemplo, los nombres de los autores serán todos abreviados o mantenidos completos simultáneamente; algo similar para el uso de la fuente itálica en los títulos de artículos o libros, etc.

Facilidad para poder cambiar el formato de las citas en el texto y la ordenación de la lista bibliográfica. Por ejemplo, podremos elegir el orden alfabético, el orden de aparición en el texto, etc., con sólo cambiar una línea en nuestro documento `LATEX`.

Economía en el mantenimiento de nuestras referencias bibliográficas. Una base de datos creada para `BIBTEX` podrá ser utilizada en tantos documentos `LATEX` como queramos.

Todo lo aquí expuesto es aplicable a versiones recientes del programa de Oren Patashnik que soporta la utilización caracteres codificados con 8 bits⁸. La versión del programa desarrollada por Niel Kempson y Alejandro Aguilar-Sierra [31], conocida con el nombre de `BIBTEX8`, tiene una capacidad de 32 bits y también soporta código de 8 bits y ficheros ASCII de configuración para determinar el orden alfabético en diferentes idiomas. Su funcionamiento es, en temas de alfabetización, actualmente, más satisfactorio que el del programa de Oren Patashnik en el que está basado.

Una base de datos para `BIBTEX` es un fichero `BIB` en formato ASCII en el que se utilizan determinados convenios para indicar lo que va a ser un *registro* (libro, artículo, etc.) de la base

⁸Con la versión que hemos testado, 0.99c, ha sido necesario a veces usar acentos en formato de 7 bits que utiliza `TeX` (véase el apartado PARA SABER MÁS del capítulo 1) para obtener una alfabetización correcta, y, aún así, la eñe se ha situado al comienzo. Está anunciada una nueva versión del programa con mayor capacidad (32 bits) y soporte para diferentes idiomas.

```

@UNPUBLISHED{carlisle,
    author = {Carlisle, David},
    year = 1995,
    title = {The {I}f {T}hen package v1.01},
    note={Documentación contenida en ifthen.doc}
@BOOK{companion,
    author={Goossens, Michel and Mittelbach, Frank and Samarin, Alexander},
    title={The { \LaTeX\ } Companion},
    publisher={Addison-Wesley},
    year= 1994}
@ARTICLE{Babel,
    author={Johannes Braams},
    title={Babel, a multilingual style-option system for use with ...}

```

Figura 2.4: Fragmento de base de datos para BIBTEX

de datos y cuáles son los *campos* (autor, título, año, etc.) dentro de los registros. La figura 2.4 muestra un fragmento de una base de datos; las entradas @ARTICLE y @BOOK indican el comienzo de un *registro* y, dentro de cada registro las entradas author, title, etc., indican el comienzo de un *campo*. Las palabras carlisle, companion, etc., que aparecen inmediatamente después de la primera llave ‘{’ que inicia un registro, son las *Etiquetas* que identifican el registro y que después serán utilizadas en el argumento del comando \cite. En esta sección suponemos que ya tenemos construida la base de datos de bibliografía; posponemos hasta la sección siguiente la descripción completa de los registros y campos que BIBTEX entiende.

Si nuestro documento principal es MiDoc.tex y nuestra base de datos de bibliografía es MiBase.bib, para producir de forma automática las citas bibliográficas con LATEX y BIBTEX es suficiente seguir los cinco pasos que se indican a continuación.

1. Incluir en nuestro documento MiDoc.tex los comandos \cite{Etiqueta} donde queramos hacer referencia a algún documento contenido en MiBase.bib. *Etiqueta* es el identificador utilizado para cada registro (registros diferentes no pueden tener la misma *Etiqueta*). La utilización en el texto del comando

\nocite{Etiqueta}

produce la entrada del contenido del registro identificado por *Etiqueta* en la lista de bibliografía sin imprimir una referencia a ella. En el argumento de \nocite se pueden utilizar varias etiquetas separadas entre sí por comas, o bien \nocite{*} para producir el listado completo de los registros contenidos en la base de datos.

2. Incluir en el documento, en el lugar en que deseemos que aparezca la bibliografía, las líneas

\bibliography{NombreBase}

\bibliographystyle{Estilo}

El argumento *NombreBase* es el nombre de la base de datos para BIBTEX que queremos utilizar; ha de tener extensión bib. Por ejemplo, si queremos utilizar MiBase.bib sustituiremos *NombreBase* por MiBase ¡pero sin la extensión bib! Debemos asegurarnos de que el sistema operativo es capaz de encontrar el fichero MiBase.bib; en caso de duda debemos indicar delante de MiBase.bib el camino adecuado. Pueden utilizarse simultáneamente va-

rias bases de datos, para lo cual basta con introducir los nombres de éstas, separados entre sí por comas.

Estilo es uno de los estilos que BIBTEX utilizará para configurar la presentación de la lista de bibliografía. Los ficheros de estilo de la bibliografía tienen extensión bst, que no hay que incluir en el argumento *Estilo*. Relacionamos a continuación los estilos estándar de BIBTEX y sus características. Pero hay muchos más; en la figura 2.6 de la página 257, ofrecemos una lista amplia de estilos para BIBTEX que aparecen en CTAN, y que han sido desarrollados por distintas organizaciones y personas ajustándose a normas de editoriales y revistas particulares o a sus propios gustos. Los estilos estándar son los siguientes:

`plain` Las entradas se ordenan alfabéticamente y se numeran (véase el ejemplo 2.13 en la página 254). El orden es autor, después año y, por último, título.

`unsrt` Similar a `plain` salvo que las entradas aparecen en la lista según el orden en el que aparecen citadas en el documento fuente.

`alpha` En vez de numerar las entradas como en el estilo `plain`, se las identifica con parte del nombre del autor y del año de publicación (véase el ejemplo 2.14 en la página 254).

El orden ahora se obtiene por etiqueta, autor, año y título.

`abbrv` Lo mismo que `plain` excepto que las entradas son más breves porque los nombres de autores, meses y revistas se abrevian (véase el ejemplo 2.15 en la página 255).

3. Compilar el documento `MiDoc.tex`. Una vez utilizados los comandos anteriores la compilación de `MiDoc.tex` hace que en el fichero `MiDoc.aux` se guarde la información correspondiente a las citas realizadas en el documento, junto con el nombre del fichero de base de datos para BIBTEX que queremos utilizar y con el nombre del estilo que hemos elegido para presentar la bibliografía.
4. Generar la lista de bibliografía. Para procesar la información referente a citas bibliográficas guardada en el fichero `MiDoc.aux` y producir el entorno `thebibliography` para LATEX, hemos de ejecutar el programa BIBTEX pasándole como argumento `MiDoc.tex`, es decir,

`bibtex MiDoc`

Esto produce dos nuevos ficheros llamados `MiDoc.bbl` y `MiDoc.blg`. El primero de los ficheros contiene el entorno `thebibliography` que se ha producido a partir de los comandos `\cite` y `\nocite` incluidos en el texto y el segundo contiene la información sobre los errores producidos durante la ejecución de BIBTEX. La información en el fichero `MiDoc.bbl` está agrupada en bloques, tales como autor o título, los cuales se separan dentro de un mismo `\bibitem` por un comando `\newblock` (véase la página 255).

5. Volver a compilar el documento `MiDoc.tex`. En esta compilación se utilizan los datos guardados en `MiDoc.bbl` dentro del entorno `thebibliography` junto con los datos de `MiDoc.aux` para escribir la lista de bibliografía y establecer las oportunas citas en el texto principal del documento.

Los ejemplos 2.13 a 2.16 muestran el uso de los distintos estilos estándar de BIBTEX enumerados en el punto 2 anterior. De un ejemplo a otro lo único que cambia es la línea en la que seleccionamos el estilo de BIBTEX (por ello en algunos casos hemos acortado el ejemplo). Es este

estilo el responsable del formato de la lista bibliográfica, de que esté o no ordenada, de las itálicas, de las comillas en los títulos, de que el nombre de pila aparezca o no abreviado, etc.

Con el estilo `plain` las entradas correspondientes a títulos de libros (registros de tipo `book`) utilizan letra itálica, mientras que los títulos correspondientes a tipos de registro distintos de `book` se cambian a minúsculas, salvo la letra inicial. Por ejemplo, en la figura 2.4 el campo

```
TITLE = {The {I}f {T}hen package v1.01},
```

correspondiente a la entrada `carlisle` contiene llaves '{' y '}' para delimitar `{I}` y `{T}` que queremos que aparezcan en mayúsculas en el título de la referencia.

EJEMPLO 2.13

```
\documentclass{book} ...
\begin{document}
... relacionada con \TeX{} y \LaTeX{}
... como \cite{textbook, lamport} y
\cite{companion}, podemos encontrar...
más especializados como \cite{metabook}...

\nocite{carlisle}
\bibliography{MiBase}\bibliographystyle{plain}
\end{document}
```

La literatura relacionada con `\TeX` y `\LaTeX` es muy abundante. Junto a libros de ámbito general, como [3,5] y [2], podemos encontrar otros libros más especializados como [4]...

Bibliografía

- [1] David Carlisle, *The If...*, 1995...
- [2] Michel Goossens, Frank Mittelbach...
- [3] Donald E. Knuth. *The \TeXbook*...
- [4] Donald E. Knuth. *The METAFONT book*...
- [5] Leslie Lamport. *\LaTeX – A document...*
- [6] Frank Mittelbach, An environment for multicolumn...

Formato de la bibliografía con el estilo `plain`

Si en vez de utilizar el estilo `plain` utilizamos el estilo `alpha`, las referencias se hacen por medio de cadenas de caracteres obtenidas a partir del apellido del autor (o editor) de la publicación, más las dos últimas cifras del año de publicación, como en [Car95], o tomando las iniciales de los nombres de los autores si hay varios, como en la entrada [GMS94].

EJEMPLO 2.14

```
\documentclass{book} ...
\begin{document}
... relacionada con \TeX{} y \LaTeX{}
... como \cite{textbook, lamport} y
\cite{companion}, podemos encontrar ...
más especializados como \cite{metabook}...

\nocite{carlisle}
\bibliography{MiBase}\bibliographystyle{alpha}
\end{document}
```

La literatura relacionada con `\TeX` y `\LaTeX` es muy abundante. Junto a libros de ámbito general, como [Knu86a, Lam85] y [GMS94], podemos encontrar otros libros más especializados como [Knu86b]...

Bibliografía

- | | |
|----------|---|
| [Car95] | David Carlisle, <i>The If...</i> , 1995... |
| [GMS94] | Michel Goossens, Frank Mittelbach... |
| [Knu86a] | Donald E. Knuth. <i>The \TeXbook</i> ... |
| [Knu86b] | Donald E. Knuth. <i>The METAFONT book</i> ... |

Formato de bibliografía con el estilo `alpha`

Los estilos `unsrt` y `abbrv` producen la lista de bibliografía en forma similar al estilo `plain` con las siguientes diferencias: con `unsrt` el orden de la lista no es el alfabético sino el de aparición en el texto; con `abbrv` algunas entradas aparecen abreviadas.

EJEMPLO 2.15

```
\documentclass{book} ...
\begin{document}
... relacionada con \TeX{} y \LaTeX{}
... como \cite{textbook, lamport} y
\cite{companion}, podemos encontrar ...
más especializados como \cite{metabook}...
\bibliography{MiBase}\bibliographystyle{abbrv}
\end{document}
```

Formato de bibliografía con el estilo abrv

La literatura relacionada con \TeX y \LaTeX es muy abundante. Junto a libros de ámbito general, como [3,5] y [2], ...

Bibliografía

- [1] D. Carlisle, *The If...*, 1995...
- [2] M. Goossens, F. Mittelbach...
- [3] D. E. Knuth. *The \TeXbook...*
- [4] D. E. Knuth. *The MetaFont book...*

La figura 2.5 refleja parte del contenido del fichero BBL obtenido en el último ejemplo 2.15. Allí aparecen los comandos `\newblock` entre bloques distintos de una misma entrada: un bloque está constituido por el nombre de los autores, otro por el título de la obra, etc.

Por defecto, los comandos

`\newblock`

son ignorados por los estilos estándar tal y como aparece ilustrado en los ejemplos 2.13, 2.14 y 2.15. Cuando esto sucede y los diversos bloques se componen seguidos se dice que el formato para la bibliografía es «cerrado». Esta opción de formato cerrado puede cambiarse a «abierto» utilizando la opción `openbib` en `\documentclass`, obteniéndose ahora que cada bloque empieza una línea nueva que ha sido sangrada la longitud

`\bibindent`

cuyo valor, si no se modifica, es 1,5 em.

Una ilustración del formato abierto aparece reflejada ejemplo 2.16. En este caso se ha hecho en combinación con el estilo `abrv`, pero puede hacerse lo mismo con cualquier otro estilo de bibliografía.

EJEMPLO 2.16

```
%%% Documento MiDoc.tex %%%%
\documentclass[openbib]{book}
\begin{document}
... relacionada con \TeX{} y \LaTeX{}
... como \cite{textbook, lamport} y
\cite{companion}, podemos encontrar ...
más especializados como \cite{metabook}...
\nocite{carlisle}
\bibliography{MiBase}\bibliographystyle{abbrv}
\end{document}
```

La literatura relacionada con \TeX y \LaTeX es muy abundante. Junto a libros de ámbito general, como [4,6] y [3], ...

Bibliografía

- [1] D. Carlisle.
The If Then package v1.01.
Documentación contenida en ifthen.doc, 1995.
- [3] M. Goossens, F. Mittelback, and A. Samarin.
The \LaTeX Companion.
Addison-Wesley, 1994.

Formato de bibliografía con el estilo `abrv` utilizado con la opción `openbib`

La gran variedad de estilos existentes (figura 2.6) pone de relieve que no sólo personas aisladas, sino colectivos enteros (químicos, informáticos, psicólogos...) tradicionalmente utilizan un formato de bibliografía que no se consigue con los cuatro estilos estándar. Cada cual deberá bus-

```
\begin{thebibliography}{1}
\bibitem{carlisle}
D.~Carlisle.
\newblock The {I}f {T}hen package v1.01.
\newblock Documentación contenida en ifthen.doc, 1995.

\bibitem{companion}
M.~Goossens, F.~Mittelbach, and A.~Samarin.
\newblock {\em The \LaTeX\ Companion}.
\newblock Addison-Wesley, 1994.
\end{thebibliography}
```

Figura 2.5: Fragmento de un fichero BBL obtenido al ejecutar BIBTEX

car aquel que responda mejor a sus costumbres. La experiencia que hemos adquirido con nuestros alumnos es que aquellos que necesitaban un formato diferente solían encontrar en el estilo chicago, eventualmente con ligeras modificaciones, la respuesta a sus necesidades.

EJEMPLO 2.17

```
\documentclass{book}
\usepackage{achicago}
\begin{document}
... relacionada con \TeX{} y \LaTeX{}
... como \cite{texbook,lampert}
\cite{companion}, podemos encontrar...
más especializados como \cite{metabook}...

\bibliography{MiBase}
\bibliographystyle{achicago}
\end{document}
```

La literatura relacionada con \TeX y \LaTeX es muy abundante. Junto a libros de ámbito general, como (Knuth 1986a; Lampert 1994) y (Goossens, Mittelbach, and Samarin 1994), podemos encontrar otros ... como (Knuth 1986b)...

Bibliografía

Carlisle, David. 1995. "The If Then package". Documentación contenida en ifthen.doc.
 Goossens, Michel, Frank Mittelbach, and Alexander Samarin. 1994. *The LATEX Companion*. Addison-Wesley.
 Knuth, Donald Ervin. 1986a. *The TeXbook*. Addison-Wesley.
 ———. 1986a. *The METAFONT book*. Addison-Wesley.

Formato de bibliografía con el estilo achicago

El estilo de bibliografía chicago resulta un tanto confuso por la existencia en CTAN de tres versiones diferentes: *chicago*, *chicagoa* y *achicago*, que producen formatos muy parecidos. La última es la de fecha más recientemente y está desarrollada por Matt Swift [64]. El estilo de bibliografía *achicago* requiere que esté cargado el paquete *achicago* que describimos en la sección 2.8.3.

2.8.2. Bases de datos para BIBTEX

Un fichero de base de datos para BIBTEX es un fichero de texto que contiene *registros*, que a su vez contienen *campos*, similares a los de la figura 2.4. Un *registro* empieza por '@' seguido de una cadena que identifica el tipo, luego una llave '{' que indica el principio del mismo y una llave '}' que indica su final; dentro de las llaves los *campos* van separados por comas. Un campo consiste en un nombre, un carácter '=' y, a continuación, su contenido. El contenido de un campo

Estilo	Descripción
abbrv.bst	Estilo estándar de BIBTeX
abstract.bst	Modificación del estilo <i>alpha</i> para incorporar abstracts.
achemso.bst	American Chemical Society.
acm.bst	Association for Computing Machinery
aer.bst	American Economic Review
aer.sty	Estilo LATEX utilizado por aer.bst
agsm.bst	Australian Government Publications
alpha.bst	Estilo estándar para BIBTeX
amsalpha.bst	Similar al estilo <i>alpha</i> creado por la American Mathematical Society, A.M.S.
amspplain.bst	Similar al estilo <i>plain</i> creado por la A.M.S.
annotate.bst	Modificación sobre el estilo <i>alpha</i> que incorpora el campo <code>annote</code>
annotation.bst	Otra modificación sobre el estilo <i>alpha</i> que incorpora el campo <code>annote</code>
apa.bst	American Psychology Association
apalike.bst	Similar al estilo <i>apa</i>
apalike2.bst	Variante del estilo <i>apalike</i>
apasoft.bst	Variante del estilo <i>apalike</i>
astron.bst	Estilo para astronomía
authordate1-4.bst	Estilos que producen la lista de referencias en el formato autor-fecha
authordate1-4.sty	Paquetes para ser utilizados con authordate1-4.bst
bbs.bst	Behavioral and Brain Sciences
cbe.bst	Council of Biology Editors
cea.bst	Computers and Electronics in Agriculture
chicago.bst	13. ^a edición del <i>Chicago Manual of Style</i>
chicago.sty	Estilo de LATEX utilizado por chicago.bst
dcu.bst	Design Computing Unit
harvard.sty	Estilo de LATEX utilizado por los estilos Harvard de bibliografía (agsm.bst, dcu.bst, kluwer.bst)
humanbio.bst	Human Biology
humannat.bst	Human Nature and American Anthropologist
is-abbrv.bst	Estilo <i>abbrv</i> con entradas ISSN y ISBN
is-alpha.bst	Estilo <i>alpha</i> con entradas ISSN y ISBN
is-plain.bst	Estilo <i>plain</i> con entradas ISSN y ISBN
is-unrt.bst	Estilo <i>unrt</i> con entradas ISSN y ISBN
jas99.bst	Journal of Atmospheric Science, Journal of Applied Meteorology, Monthly Weather Review
jmb.bst	Journal of Molecular Biology
jmb.sty	Estilo LATEX utilizado por jbact.bst
jtb.bst	Journal of Theoretical Biology
kluwer.bst	Kluwer Academic Publishers
nar.bst	Nucleic Acid Research
nar.sty	Estilo LATEX utilizado por nar.bst
natbib.bst	Estilo genérico que implementa varios estilos con referencias «autor-año»
natbib.sty	Estilo LATEX utilizado por natbib.bst
newapa.bst	Modificación de apalike.bst
newapa.sty	Estilo LATEX utilizado por newapa.bst
phcip.bst	American Institute of Physics
phcpc.bst	Computer Physics Communications
phiae.abst	Conference of the International Atomic Energy Agency
phjcp.bst	Journal of Computational Physics
phnf.bst	Nuclear Fusion
phnleft.bst	Nuclear Fusion Letters
phpf.bst	Physics Fluids
phppcf.bst	Versión del estilo <i>apalike</i> usado en varias revistas de Física
phreport.bst	Internal Physics Report
phrmp.bst	Reviews of Modern Physics
plain.bst	Estilo estándar para BIBTeX
plainyr.bst	Similar a <i>plain</i> , ordenando primero por año
siam.bst	Society of Industrial and Applied Mathematics
unsrt.bst	Estilo estándar para BIBTeX

Figura 2.6: Lista de algunos estilos para BIBTeX que aparecen en CTAN

es una cadena de caracteres delimitada por un par de comillas, ", o por llaves, '{' al principio y '}' al final. BIBTEX no distingue entre mayúsculas y minúsculas en los nombres de los *registros* y de los *campos*, y las comillas o llaves pueden ser omitidas para campos numéricos como, por ejemplo, *year*. Así, el primer registro de la figura 2.4 podría haber sido introducido como sigue:

```
@UNPUBLISHED{carlisle,
    AUTHOR = "Carlisle, David",
    YEAR = "1995",
    TITLE = "The {I}f {T}hen package v1.01",
    NOTE = "Documentación contenida en ifthen.doc"}
```

Como regla general, para escribir un fichero de base de datos para BIBTEX, se debe tener presente que es el estilo de bibliografía el que determina cómo se imprimen los nombres de los autores (abreviados o no), títulos de los artículos (cómo se utilizan las mayúsculas) o cuándo se cambia, por ejemplo, el tipo de letra, y todo ello buscando estandarizar la forma de escribir las referencias.

Al introducir un registro en la base de datos, la primera cosa a decidir es su tipo. BIBTEX suministra suficientes tipos de registro para satisfacer casi todas las necesidades. Registros de tipo diferente pueden contener campos diferentes. Por ejemplo, un registro con los datos de un artículo en una revista podría contener el volumen y el número de la revista, que no tiene sentido normalmente para un libro. Los campos de un registro se dividen en tres clases.

Campos necesarios. Si se omite alguno de los campos que tienen esta consideración para un tipo de registro, se produce un mensaje de advertencia al ejecutar BIBTEX. En general, eso es indicativo de que se está utilizando un tipo de registro inadecuado.

Campos optativos. La información contenida en los campos de esta clase se utilizará si está presente, pero puede ser omitida sin causar problemas.

Campos ignorados. De esta clase forman parte el resto de los campos de un registro. BIBTEX los ignora, pero pueden ser utilizados para poner información complementaria en el fichero BIB; por ejemplo, se puede incluir un resumen de un artículo utilizando el campo *abstract* del registro, y aunque los estilos estándar no extraigan esa información, siempre será posible construir un fichero de estilo personal que lo haga.

Los campos pueden contener (al menos para las versiones actuales de BIBTEX) indistintamente caracteres en formato 7 bits, con el formato de acentos de TEX (véase el apartado PARA SABER MÁS del capítulo 1), o bien caracteres acentuados usando la página de códigos de 8 bits. También pueden contener comandos de LATEX (lo que puede ser de interés especialmente en los campos optativos o en bases de datos como las consideradas en la página 264), pero conviene tener en cuenta que BIBTEX no respeta el formato de las líneas existentes en la base de datos, debido a lo cual no deben incluirse comentarios (que comienzan con un %) o líneas en blanco (que deben ser sustituidas por comandos \par). Además, el número de caracteres que pueden incluirse en un campo está limitado. Esta limitación, obviamente, no es significativa en la mayor parte de los campos, pero puede serlo para alguno en especial (por ejemplo, *abstract*); la forma de eludirla es utilizar comandos \input.

La relación de registros y campos que sigue recoge todos los registros y campos reconocidos por los estilos estándar de bibliografía para BIBTEX, es decir, plain, unsrt, alpha, abrv.

Campos

La lista de campos que sigue cubre un amplio abanico de necesidades; sin embargo, a veces no deben tomarse demasiado en serio los nombres dados a los mismos y podemos incluir en ellos la información que a nosotros nos parezca relevante.

address Habitualmente la dirección del publisher u otro tipo de institución. En [39] se recomienda utilizar este campo sólo para pequeñas editoriales, pues puede en este caso ayudar al lector a localizar la publicación. Para grandes editoriales puede dejarse vacío.

annote Una nota larga utilizada para las listas de bibliografía comentadas. No es utilizado por los estilos estándar.

author Los nombres de los autores que deben introducirse conforme a las reglas explicadas en la página 260.

booktitle Título de un libro, parte del cual está siendo citado. Para registros tipo book debe utilizarse el campo **title**.

chapter Un número de capítulo, sección, etc.

crossref Un campo «técnico» para referencias cruzadas entre registros, véase la página 263.

edition La edición de un libro. Debería utilizarse un ordinal empezando por mayúsculas.

editor Nombres de los editores. Si existe también un campo **author** en el registro, entonces el campo **editor** puede utilizarse para recoger el nombre del editor del libro o colección donde la referencia aparece.

howpublished Cómo ha sido publicado algo que no responde a las publicaciones estándar.

institution La institución que financia un informe técnico.

journal El nombre de una revista. Las abreviaturas comúnmente aceptadas para nombres de revistas suelen estar disponibles en publicaciones especializadas, y muchas veces están disponibles en ficheros electrónicos de dominio público.

key Utilizado para referencias cruzadas y para crear una etiqueta cuando **author** es vacío.

month El mes en el que un trabajo fue publicado o escrito.

note Cualquier información adicional que pueda ayudar al lector.

number El número de una revista, informe técnico o trabajo en una serie. Habitualmente las revistas se identifican por su volumen y número; algunas veces también libros cuando forman parte de una serie.

organization La organización que financia una conferencia o que publica un manual.

pages Una o más páginas o un intervalo para las mismas. Para mantener compatibilidad con las bases de datos *Scribe*, los estilos estándar convierten un guión sencillo (7-23) en uno doble que es el utilizado en *LAT_EX* (7–23) para denotar intervalos de páginas. Se debe utilizar ‘+’ para indicar desde una página en adelante.

publisher Nombre de quien publica el documento referenciado.

school Nombre de la escuela, facultad o instituto donde ha sido escrita una tesis.

series Nombre de una serie o conjunto de libros. Cuando se cita un libro completo el campo **title** corresponde a su título y el campo **series** corresponde a la serie o multivolumen donde el libro ha sido publicado.

title El título de un trabajo, libro, etc.

type El tipo de informe técnico.

volume El volumen de una revista o volumen de un libro que consta de varios volúmenes.

year El año de publicación o, para un trabajo no publicado, el año en el que fue escrito. Generalmente se deben escribir las cuatro cifras del año. No obstante, los estilos estándar pueden manejar cualquier campo **year** cuyos últimos cuatro caracteres sean numéricos.

Sobre el contenido de los campos author y editor. Dentro de los campos **author** y **editor**, los nombres deben escribirse completos, aunque sólo vayamos a utilizar abreviaturas de los nombres de pila, o, en su caso, como aparecen en el documento que se cita. Si en el campo hay que incluir varios autores, éstos deben separarse entre sí mediante la partícula

and

Será el estilo utilizado por BIBTEX el que en última instancia decidirá cómo se escribirán los nombres de autores y editores. En general, las entradas correspondientes a autores serán del tipo

author = {Sandoval, Juan}

si bien la entrada

author = {Juan Sandoval}

es igualmente aceptable. La primera es la forma recomendable de introducir los nombres cuando éstos utilizan más de un apellido, como ocurre con nombres españoles. Por ejemplo, es sumamente aconsejable escribir

"Sandoval Inocente, Juan"

puesto que en caso de haber introducido el nombre como

"Juan Sandoval Inocente"

BIBTEX hubiera considerado a Sandoval como el «middle name» del mundo anglosajón y no como apellido. En general los nombres para BIBTEX tienen cuatro partes que, eventualmente, pueden ser vacías, y que son denotadas por **First**, **von**, **Last** y **Jr**. BIBTEX admite tres posibles formas para nombres:

"First von Last" "von Last, First" "von Last, Jr, First"

Si queremos agrupar varias palabras como una sola parte, deben ponerse entre llaves. Así,

"van der Vaerden, Juan"

tiene las partes **von**, **Last** y **First**, mientras que

"{van der Vaerden}, Juan"

sólo tiene parte **Last** y **First**. En nombres como

"Juan {\MakeUppercase{d}e La\ } Torre"

BIBTEX reconocerá **De La** como la parte **von** del nombre, y si, dependiendo del estilo, tiene que abbreviar el nombre escribirá *J. De La Torre* y no *J. D. L. Torre*. Por otro lado, BIBTEX también maneja nombres separados por guiones como *Jean-Paul Sartre* que será abreviado en su caso como *J.-P. Sartre*.

Mayúsculas y minúsculas en los títulos. El estilo de bibliografía determina cuándo se utilizan mayúsculas o minúsculas en los *títulos*. Así, por ejemplo, en el título de un artículo el estilo cambiará a minúsculas cualquier mayúscula que no corresponda al principio del título. Si se quiere que otros caracteres aparezca tal cual están en el registro deberán ser encerrados entre llaves.

Registros

Los registros para una base de datos de BIBTEX se inician con un carácter ‘@’. La lista que sigue contiene los catorce tipos reconocidos por los estilos estándar.

[article] Un artículo de una revista.

Campos necesarios: author, title, journal, year.

Campos optativos: volume, number, pages, month, note.

[book] Un libro con una editorial especificada.

Necesarios: author o editor, title, publisher, year.

Optativos: volume o number, series, address, edition, month, note.

[booklet] Un trabajo que es imprimido y distribuido sin el nombre de una editorial o una institución que lo financia.

Necesarios: title.

Optativos: author, howpublished, address, month, year, note.

[conference] Lo mismo que inproceedings, incluido para compatibilidad con *Scribe*.

[inbook] Una parte de un libro, que puede ser un capítulo (o sección, etc.) o intervalo de páginas.

Necesarios: author o editor, title, chapter o pages, publisher, year.

Optativos: volume o number, series, type, address, edition, month, note.

[incollection] Una parte de un libro que tiene su propio título.

Necesarios: author, title, booktitle, publisher, year.

Optativos: volume o number, series, type, chapter, pages, address, edition, month, note.

[inproceedings] Un artículo publicado en las actas de un congreso.

Necesarios: author, title, booktitle, year.

Optativos: editor, volume o number, series, pages, address, month, publisher, organization, note.

[manual] Documentación técnica.

Necesarios: title.

Optativos: author, organization, address, edition, month, year, note.

[masterthesis] Tesina de licenciatura.

Necesarios: author, title, school, year.

Optativos: type, address, month, note.

[misc] Para cualquier referencia que no encaje en los demás tipos.

Necesarios: ninguno.

Optativos: author, title, howpublished, month, year, note.

phdthesis Tesis doctoral.

Necesarios: author, title, school, year.

Optativos: type, address, month, note.

proceedings Las actas de un congreso.

Necesarios: title, year.

Optativos: editor, volume o number, series, address, month, note, publisher, organization.

techreport Un informe publicado por una escuela u otra institución, habitualmente numerado dentro de una serie.

Necesarios: author, title, institution, year.

Optativos: type, number, address, month, note.

unpublished Un documento que tiene un autor y un título, pero que no ha sido publicado formalmente.

Necesarios: author, title, note.

Optativos: month, year.

Abreviaturas

BIBTeX puede trabajar con abreviaturas para los campos. Por ejemplo, el estilo *plain* utiliza abreviaturas para los meses en inglés. Podemos también definir nuestras propias abreviaturas utilizando para este fin la función @STRING. Por ejemplo, las líneas que siguen, incorporadas a nuestro fichero BIB, definen las abreviaturas PN y CLMPS para su uso posterior. Tal y como se muestra debajo se pueden concatenar las cadenas definidas mediante el símbolo #.

```
@STRING{PN = "Pearson"}
@STRING{E = "España"}
@STRING{CLMPS = "Cascales, Bernardo and Lucas, Pascual and Mira, José
               Manuel and Pallarés, Antonio and Sánchez-Pedreño, Salvador"}
@BOOK{CLMPS2,
       author=CLMPS,
       title={El libro de {\LaTeX}},
       publisher=PN # E,
       year= 2003}
```

Algunas organizaciones han creado, y actualizan, ficheros con abreviaturas para revistas especializadas, pretendiendo con ello estandarizar la forma en la que se realizan las referencias a estas revistas. Así, por ejemplo, la *American Mathematical Society* distribuye un fichero llamado *mrabbrev.bib* que contiene abreviaturas para todas las revistas de matemáticas alguna vez citadas en *Mathematical Reviews*. Si queremos utilizar las abreviaturas de *mrabbrev.bib* en nuestra base *Mibase.bib* será suficiente que el documento *MiDoc.tex* que estamos procesando contenga la línea

```
\bibliography{mrabbrev,MiBase}.
```

Es, desde luego, aconsejable, siempre que se pueda, utilizar abreviaturas para nombres de revistas, editoriales, etc., ampliamente aceptadas.

Referencias entre registros

BIBTEX incorpora el campo `crossref` para posibilitar cierto tipo de referencias cruzadas dentro de las citas bibliográficas.

Imaginemos que el fichero con nuestra base de datos contiene

```
@INPROCEEDINGS{lol,  
    crossref = "pep",  
    author = "Grandes, Lola",  
    title = "El ocultismo sale a la luz",  
    pages = "120-125"}  
@PROCEEDINGS{pep,  
    editor = "Salado, Pepe",  
    title = "Ciencias ocultas en la oscuridad, 1999",  
    booktitle = "Ciencias ocultas en la oscuridad, 1999"}
```

Supongamos ahora que en nuestro documento hemos incluido `\cite{lol}`. Después de la compilación con LATEX, al ejecutar BIBTEX suceden dos cosas. La presencia de `crossref` hace que la entrada `lol` tome de la entrada `pep` todos los campos que no tiene y que, en este caso, son `editor` y `booktitle`. Observe que, en estilos estándar al menos, el campo `booktitle` es irrelevante para un registro tipo PROCEEDINGS. El campo `booktitle` aparece aquí para proveer de este campo a los registros que mediante una referencia cruzada puedan precisarlo. No importa cuántos artículos presentados en el congreso «*Ciencias ocultas en la oscuridad, 1999*» haya en nuestra base de datos; si utilizamos `crossref` el campo `booktitle` correspondiente aparecerá sólo una vez en la base de datos, contribuyendo así, entre otras cosas, a que el título del libro aparezca exactamente igual en todos los artículos. La segunda cosa que ocurre en presencia de `crossref` es que BIBTEX automáticamente pone la entrada `pep` en la lista de referencias si es llamada al menos dos veces por otras referencias, vía el campo `crossref`, con `\cite` o `\nocite`, incluso aunque no se cite explícitamente la entrada `pep`. Para garantizar que este esquema funcione adecuadamente, un registro que va a ser llamado vía `crossref` debe estar en la base de datos después del registro que va a hacer referencia a él. De esta forma, es una buena idea poner todos los registros que van a ser referenciados al final de la base de datos.

Los estilos estándar ofrecen la posibilidad de utilizar el campo `crossref` para referencias cruzadas en las siguientes situaciones:

- un registro ARTICLE puede hacer referencia a otro registro ARTICLE, BOOK o INBOOK;
- un registro INCOLLECTION puede hacer referencia a un registro BOOK;
- un registro INPROCEEDINGS puede hacer referencia a un registro PROCEEDINGS.

Como comentario final a las referencias cruzadas dentro de un fichero de bibliografía, debemos decir que es posible incluir comandos `\cite` en campos de la base de datos, lo que no guarda relación alguna con la funcionalidad del campo `crossref` comentado hasta ahora. Así, por ejemplo, un campo

`note = "Este articulo fue el más aclamado de \cite{pep}"`

puede ser útil para mantener referencias dentro de nuestra propia base de datos.

Definiciones en preámbulos

Existe una función @PREAMBLE análoga a la función @STRING, salvo que ahora no existe nombre para la cadena ni signo igual. Los estilos estándar escriben literalmente en el fichero BBL, inmediatamente antes del entorno `thebibliography`, lo que se ha incluido en el argumento de @PREAMBLE, de forma que L^AT_EX puede interpretar el texto o comandos que hayamos incluido allí. Por ejemplo, el estilo *alpha* usa un superíndice ⁺ para representar los nombres omitidos a la hora de construir una referencia. Es posible cambiar ⁺ por cualquier otra cosa que sea más de nuestro agrado modificando el valor del comando

```
\etalchar
```

Si queremos mantener esta modificación en todos los documentos que utilicen nuestro fichero BIB es una buena opción incluir en él la línea

```
@PREAMBLE{"\renewcommand*\etalchar}{\$^*\$}"}
```

con lo cual ⁺ en las omisiones será sustituido por *. Obsérvese el uso de las " rodeando lo que queremos que se escriba literalmente en el preámbulo.

Otras bases de datos para BIBT_EX

Aunque el programa BIBT_EX ha sido diseñado para ser utilizado con bases de datos para referencias bibliográficas, también es posible construir estilos BST y STY, para utilizar conjuntamente BIBT_EX y L^AT_EX, de forma que podamos manejar otras bases de datos, como teléfonos, direcciones, problemas...

 Los autores de este libro han desarrollado, para uso personal, una estructura de base de datos pensada para ser utilizada con problemas de matemáticas. Cada problema se escribe dentro de un nuevo tipo de registro, de nombre @problema, cuyos campos son `titulo`, `enunciado`, `claves`, `dificultad`, `referencia`, `indicacion`, `solucion` y `nota`, en lugar de los usuales `author`, `title`, etc. Al escribir los registros se utiliza la sintaxis ordinaria y todos los campos, salvo `enunciado`, son optativos. Un estilo de bibliografía `problema.bst` y un paquete `problema` permiten realizar listados (hojas de problemas) de forma sencilla mediante un comando `\nocite` en cuyo argumento se incluyen las etiquetas de los problemas que deseamos listar. El control de los campos que queremos que aparezcan en el listado se realiza mediante opciones al cargar el paquete `problema`; los nombres de tales opciones coinciden con los nombres de los campos y la opción `enunciado` siempre está activa; una opción adicional, de nombre `todos`, incorpora para cada registro todos los campos que no estén vacíos.

Con las mismas ideas es fácil realizar estructuras de bases de datos y paquetes para ser utilizados en correo personalizado.

Creación y mantenimiento de bases de datos para BIBT_EX

L^AT_EX y BIBT_EX forman un buen tandem para producir listas de bibliografía en un documento. Sin embargo, crear una base de datos para BIBT_EX utilizando un editor de texto y confiando la estructura de la misma a nuestra memoria no es tan atractivo: téngase en cuenta que las etiquetas

no pueden repetirse y que para los catorce tipos de registro de las bases de datos siempre deben estar presentes, al menos, los campos necesarios. Si nuestra base de datos crece considerablemente su mantenimiento puede ser un problema serio.

Los entornos de trabajo modernos, como los indicados en la lección 2, permiten desde los menús desplegables, al menos, ir añadiendo registros a una base de datos mediante plantillas. Además existen varias herramientas gratuitas más específicas para la creación y mantenimiento de bases de datos para BIBTEX. Entre ellas nos limitamos a citar dos, disponibles en todas las plataformas, JBIBTEXMANAGER, implementado en lenguaje JAVA y TKBIBTEX, implementada en lenguaje TCL/TK; y una más, WINBIBDB, desarrollada específicamente para MS-WINDOWS. Aunque cada una tiene sus prestaciones particulares, tienen en común las siguientes características:

- ofrecen un entorno amigable que permite navegar por la base de datos, extraer registros seleccionados e introducir y editar registros de forma sencilla;
- permiten que el usuario genere y mantenga la base de datos sin conocer la sintaxis y reglas de las bases de datos de BIBTEX;
- trabajan directamente sobre ficheros de base de datos para BIBTEX creados con cualquier editor o herramienta específica;
- proporcionan herramientas de búsqueda y edición dentro de la base de datos.

Si usted escribe matemáticas y dispone de conexión a las bases de datos MathSciNet (American Mathematical Society) y Zentralblatt Math (European Mathematical Society), puede obtener de estas bases de datos el registro correspondiente a cualquier publicación que aparezca en ellas, en un formato adecuado para BIBTEX, que puede añadir a su base de datos personal.

Adaptar los estilos de bibliografía

Diseñar un estilo de bibliografía no es tarea sencilla porque requiere el conocimiento del lenguaje que BIBTEX utiliza. En nuestro libro [11] dimos algunas orientaciones para realizar esa tarea. Además, con la gran variedad de estilos existentes en CTAN no será difícil encontrar alguno que se adapte, esencialmente al menos, a nuestras necesidades. Sin embargo, hay un escollo que rara vez estará resuelto, y es que los estilos de bibliografía están desarrollados para las lenguas anglosajonas, lo que trae consigo que algunos textos que BIBTEX inserta en determinadas circunstancias están en el idioma inglés y desgraciadamente no existe todavía en BIBTEX algo con una funcionalidad similar a la que tiene *babel* en LATEX. El más significativo de tales textos es la partícula «and» que puede aparecer entre los autores cuando son varios y que en español debería ser cambiada por «y», pero hay más (and others...). Evidentemente, en los campos de la base de datos de bibliografía el separador debe continuar siendo ‘and’ (véase la página 260), pero al construir la lista bibliográfica debe aparecer «y» («y otros»...). La forma más sencilla de resolver el problema es, una vez elegido el estilo de bibliografía que mejor se adapta a nuestras necesidades, supongamos que sea plain.bst, guardar una copia del mismo con un nombre diferente, digamos miplain.bst (siempre debe hacerse así si se modifica cualquier archivo, de acuerdo con la licencia GNU) y cada vez que obtengamos un resultado no deseado buscar en el archivo miplain.bst el texto indeseado para modificarlo convenientemente. A veces será sencillo y otras no tanto porque la partícula bus-

Comando	Imprime	Ejemplo
\cite{Etiqueta}	Entre paréntesis, lista de hasta tres autores o un ‘et al.’ y el año.	(Goossens, Mittelbach, and Samarin 1994)
\citeA{Etiqueta}	Como \cite, pero sin el año.	(Goossens, Mittelbach, and Samarin)
\citeN{Etiqueta}	Como \cite, pero el paréntesis sólo afecta al año.	Goossens, Mittelbach, and Samarin (1994)
\citeyear{Etiqueta}	Como \cite, pero sin autores.	(1994)
\citeNP{Etiqueta}	Como \cite, pero sin paréntesis.	Goossens, Mittelbach, and Samarin 1994
\citeANP{Etiqueta}	Como \citeA, pero sin paréntesis.	Goossens, Mittelbach, and Samarin
\citeyearNP{Etiqueta}	Como \citeyear, pero sin paréntesis.	1994

Cuadro 2.3: Comandos del paquete *achicago*

cada tiene múltiples apariciones, alguna de las cuales pueden no ser «sólo texto» intercambiable, sino formar parte del código del estilo con función de «comandos» para BIBTEX. Con un poco de experimentación, y deshaciendo los cambios cuando se detecte un error, no resulta complicado alcanzar el objetivo marcado.

☞ Nosotros hemos realizado adaptaciones por este procedimiento en algunos estilos de bibliografía que hemos usado e incluimos en el CD-ROM tales adaptaciones, cuyo nombre coincide con el original más una partícula final *loc*, indicando que se trata de una modificación local.

2.8.3. Algunos paquetes útiles para el manejo de bibliografía

achicago El paquete *achicago* desarrollado por Matt Swift [65] implementa las recomendaciones de [14], e incorpora la lista de comandos que se indica en el cuadro 2.3. Este paquete ha sido diseñado para ser utilizado con el estilo de bibliografía *achicago.bst*. El ejemplo 2.17, página 256, ilustra la utilización de los comandos incorporados por *achicago*, supuesto que la lista de bibliografía se ha generado como se indica en el código, el cual resulta suficientemente autoexplicativo.

cite El paquete *cite*, desarrollado por Donald Arseneau [1], agrupa una lista de tres o más números para referencias consecutivas en un intervalo. Por ejemplo, [1,2,3,4,8] se transformaría en [1–4,8]. Implementa también el comando

\citen

para posibilitar que las referencias en el texto aparezcan sin los corchetes.

citesort El paquete *citesort*, desarrollado por Ian Green [25], mejora un poco el paquete anterior y ordena los números antes de agruparlos. Así, por ejemplo, una cita a [1,2,3,4,8,5,7,6] se convertiría, utilizando *citesort*, en [1–8].

footbib El paquete *footbib*, desarrollado por E. Domenjoud [17], implementa las herramientas necesarias para dar respuesta a la costumbre que existe en algunas disciplinas de escribir las referencias bibliográficas al pie de la página en la que aparecen citadas (con independencia de que se incluyan todas juntas en una lista de bibliografía) y cuando una misma referen-

cia bibliográfica es usada en páginas cercanas, en lugar de volver escribir los datos de la referencia, se utiliza una frase del tipo «*Ibidem pág...».*

Señalaremos únicamente que el paquete consigue esos resultados implementando nuevos comandos, compatibles con el comando `\cite`, entre los que destacamos los que siguen.

<code>\footcite{Etiqueta}</code>	<code>\footcite*{Etiqueta}</code>	<code>\footnocite{Etiqueta}</code>
----------------------------------	-----------------------------------	------------------------------------

El comando `\footcite` tiene el mismo funcionamiento y sintaxis que el comando `\cite` salvo que las referencias bibliográficas aparecen al pie de la página en la que se citan. El comando `\footcite*` pone la marca en el texto pero no incluye la referencia a pie de página, mientras que `\footnocite` hace aparecer la referencia al pie de página sin poner marca alguna en el texto; su funcionamiento es similar al de `\footnotemark` y `\footnotetext`, respectivamente.

<code>\footbibliography{NombreBase}</code>	<code>\footbibliographystyle{Estilo}</code>
--	---

Estos comandos se corresponden con `\bibliography` y `\bibliographystyle`, respectivamente, en sintaxis y funcionalidad.

EJEMPLO 2.18

```
\usepackage{footbib}
```

La literatura relacionada con `\TeX{}` y `\LaTeX{}` es muy abundante. Junto a libros de ámbito general%, `\footcite{texbook,lampert,companion}`, podemos encontrar otros libros más especializados`\footcite{metabook}...`

```
\bibliography{MiBase}
\bibliographystyle{plain}
\end{document}
```

Bibliografía a pie con el paquete `footbib`

La literatura relacionada con `\TeX` y `\LaTeX` es muy abundante. Junto a libros de ámbito general^[3,5,2] podemos encontrar otros libros más especializados^[4]...

-
- [3] Donald Knuth. *The TeXbook*. Addison-Wesley, 1986.
 - [5] Leslie Lamport. *\TeX —A document ...*
 - [2] M. Goossens, F. Mittelbach, and A. Samarin...
 - [4] *Ibidem* pág. 24

El paquete escribe la información relativa a las citas creadas con `\footcite` en el archivo `MiDoc.fb.aux` dejando que los comandos `\cite` utilicen `MiDoc.aux`. En consecuencia, el ejecutable `BIBTeX` debe actuar sobre `MiDoc.fb.aux` para la bibliografía a pie y sobre `MiDoc.aux` para la bibliografía ordinaria, si la hay. Cada una de estas bibliografías puede utilizar sus propias bases de datos y estilo.

Están implementadas varias formas de numerar estas citas bibliográficas a pie, de modo que la numeración se inicie cada página, capítulo, etc. El control de estos formatos se realiza mediante varias opciones al cargar el paquete; los detalles están explicados en la documentación que lo acompaña. Señalaremos que, en nuestra opinión, el resultado estético es razonablemente satisfactorio mientras no se utilicen simultáneamente notas a pie de página y citas bibliográficas a pie, pero si se utilizan ambas construcciones el resultado es, estéticamente, discutible, ya que en tal caso se crean sendas cajas apiladas en la parte inferior de la página;

la primera para las notas a pie y la segunda para la bibliografía, con sus correspondientes filetes de separación del texto que les precede.

El lector interesado en presentar las citas bibliográficas según el procedimiento que ilustra sucintamente el ejemplo 2.18 debe consultar la información que acompaña al paquete.

2.8.4. Varias listas de bibliografía

Incorporar varias listas de bibliografía puede ser especialmente útil para documentos largos, con secciones independientes, en actas de congresos conteniendo diferentes artículos o en libros con partes o capítulos diferenciados escritos por distintos autores.

EJEMPLO 2.19

```
\documentclass{article} \usepackage{bibunits}
\begin{document}
\section{Introducción}
En este artículo... el libro \cite{GMS94}...
\section{LaTeX}
\begin{bibunit}[plain]
\LaTeX{} no es únicamente un sistema para
escribir matemáticas. En los libros,
\cite{GMS94} y \cite{Hah96} se
pueden ...
\putbib[MiBase]
\end{bibunit}
\section{Herramientas de \LaTeX}
Cada \emph{paquete} es un fichero ASCII
que implementa nuevas posibilidades
para \LaTeX. Los paquetes \cite{Mit95} y
\cite{Zan93}...
\renewcommand*\refname{Referencias globales}
\bibliography{MiBase}
\bibliographystyle{alpha}
\end{document}
```

Varias listas de bibliografía con el paquete bibunits

El paquete `bibunits`, elaborado por José Alberto Fernández [19] y modificado por Thorsten Hansen [28], permite trabajar con varias listas de bibliografía para distintas unidades de un mismo documento. Las unidades pueden ser capítulos, secciones o partes delimitadas por un entorno `bibunit` definido en el paquete. Este paquete separa las citas de cada unidad de texto en un fichero diferente para poder ser procesado por `BIBTEX`. Una misma referencia puede estar en más de una lista de bibliografía. La sección global de bibliografía en el documento principal puede estar presente también.

Una vez cargado el paquete, puede utilizarse el entorno

```
\begin{bibunit}[Estilo]
...
\putbib[NombreBase]
\end{bibunit}
```

1. Introducción

En este artículo describimos ...el libro [GMS94]...

2. LATEX

`LATEX` no es únicamente un sistema para escribir matemáticas... En los libros [1] y [2] se pueden ...

Referencias

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LATEX Companion* ...

[2] Jane Hahn. *LATEX for everyone. A reference guide*...

3. Herramientas de LATEX

Cada `paquete` es un fichero ASCII que implementa nuevas posibilidades para `LATEX`. Los paquetes [Mit95] y [Zan93]...

Referencias globales

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LATEX Companion* ...

[Hah96] Jane Hahn. *LATEX for everyone. A reference*...

[Mit95] Frank Mittelbach. *An environment for multi...*

para delimitar una parte del documento en la que queremos que se incluya una lista de bibliografía independiente. En los argumentos opcionales del entorno `bibunit`, *Estilo* corresponde a uno de los estilos para BIBTEX y *NombreBase* corresponde a la base, o bases, de bibliografía para BIBTEX que queremos utilizar. Las observaciones de la página 252 referentes a *Estilo* y *NombreBase* se aplican aquí también. El ejemplo 2.19 ilustra cómo utilizar este paquete para generar una lista de bibliografía para una parte de un documento, junto con la lista general de bibliografía.

La primera compilación del documento `MiDoc.tex` del ejemplo 2.19 producirá, junto a los ficheros habituales, otro fichero llamado `bu1.aux`. Si ahora ejecutamos BIBTEX sobre `MiDoc.aux` y sobre `bu1.aux`, dos nuevas compilaciones de `MiDoc.tex` producirán las dos listas bibliográficas que aparecen en el ejemplo. Si incorporamos más unidades delimitadas por el entorno `bibunit`, tendremos un fichero AUX para cada una de ellas cuyos nombres serán `bu1.aux`, `bu2.aux`, etc. Habrá que ejecutar BIBTEX sobre cada uno de estos ficheros para obtener las listas de bibliografía de la unidad correspondiente.

El paquete `bibunits` también define el comando

```
\bibliographyunit [Unidad]
```

donde el argumento optativo *Unidad* debe reemplazarse por `\chapter` o `\section` según cuáles sean las unidades de estructura que deseamos como las partes para las que se van a generar las listas de bibliografía independientes; dentro de cada *capítulo* o *sección* el comando

```
\putbib
```

será el encargado de escribir su correspondiente lista bibliográfica, dondequiera que éste haya sido ubicado.

Siempre que `\bibliographyunit` esté activo, los estilos y bases de datos de BIBTEX para las unidades declaradas se fijan mediante los comandos

```
\bibliography*{NombreBase}
```

```
\bibliographystyle*{Estilo}
```

Los comandos `\bibliography*` y `\bibliographystyle*` fijan, a partir del momento en el que aparecen en el texto, las bases de datos y el estilo de la bibliografía que utilizarán las unidades declaradas mediante el comando `\bibliographyunit` o mediante un entorno `bibunit`. Pueden utilizarse varias veces.

Las bases de datos y el estilo para la bibliografía global se declaran mediante los comandos `\bibliography` y `\bibliographystyle`. Y también serán utilizadas por todas las unidades mientras no aparezcan comandos `\bibliography*` y `\bibliographystyle*` que cambien las bases de datos y el estilo. La utilización en un mismo documento del entorno `bibunit` y unidades declaradas mediante `\bibliographyunit` puede ocasionar efectos indeseados. Con el comando `\bibliographyunit` sin argumentos se pueden desactivar en cualquier momento las unidades declaradas a través de `\bibliographyunit [Unidad]`.

El ejemplo 2.20 ilustra la utilización de estos comandos, y en él se ha incorporado el código

```
\let\stdthebibliography\thebibliography  
\renewcommand*{\thebibliography}{%  
 \let\section\subsection\stdthebibliography}
```

para asegurar el correcto funcionamiento del paquete `bibunit`. En palabras del autor del paquete [28], esta redefinición es necesaria porque si se crean listas bibliográficas a nivel de secciones, el entorno `thebibliography` también aparece a este nivel, y la información de las unidades previas no está disponible para la generación correcta de los ficheros auxiliares.

El paquete `bibunits` también incorpora el comando

```
\cite*{Etiqueta}
```

que utilizado dentro de una unidad local de bibliografía pone la referencia marcada con *Etiqueta* en la lista de referencias de la correspondiente unidad y en la lista de referencias global, tal y como se aprecia en los ejemplos 2.19 y 2.20.

EJEMPLO 2.20

```
\documentclass{article}\usepackage{bibunits}
\let\stdthebibliography\thebibliography
\renewcommand*\{\thebibliography\}%
\let\section\subsection\stdthebibliography
\begin{document}
\bibliographyunit[\section]
\bibliographyunit
\section{Introducción}
En este artículo describimos... Entre otros,
el libro \cite{GMS94}...
\bibliographyunit[\section]
\bibliographystyle*{plain}
\bibliography*{mibase}
\section{\LaTeX{}}
\LaTeX{} no es únicamente un sistema...
En los libros,\cite{GMS94} y \cite*{Hah96}...
\putbib

\bibliographyunit
\section{Herramientas de \LaTeX{}}
Cada \emph{paquete} es un fichero ASCII que implementa nuevas po-
sibilidades para \LaTeX. Los paquetes [Mit95] y [Zan93]...
\renewcommand*\{\refname\}{Referencias Globales}
\bibliography{MiBase}
\bibliographystyle{alpha}
\end{document}
```



1. Introducción

En este artículo describimos ...Entre otros, el libro [GMS94]...

2. LATEX

LATEX no es únicamente un sistema para escribir matemáticas... En los libros [1] y [2] se pueden...

Referencias

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LATEX Companion* ...

[2] Jane Hahn. *LATEX for everyone. A reference guide*...

3. Herramientas de LATEX

Cada paquete es un fichero ASCII que implementa nuevas po-sibilidades para LATEX. Los paquetes [Mit95] y [Zan93]...

Referencias Globales

[GMS94] Michel Goossens, Frank Mittelbach, and Alexan-
der Samarin. *The LATEX Companion* ...

[Hah96] Jane Hahn. *LATEX for everyone. A reference*...

[Mit95] Frank Mittelbach. An environment for multi...

[Zan93] Timothy Van Zandt. Documentation for...

2.8.5. Glosarios con BIBTEX

Hemos visto en el apartado 2.7.8 el modo en el que pueden construirse glosarios con LATEX y MAKEINDEX. También señalamos allí que, aunque estas herramientas prestan importantes ayudas, la situación no es tan confortable como para los índices terminológicos. Pero la cuestión fundamental es que, aunque existan ciertas similitudes entre ambos, un glosario no es un índice terminológico. Ciertamente en ambos casos se pretende extraer «determinados conceptos» del texto y elaborar con ellos una lista alfabética, pero en el glosario se persigue además hacer una descripción «relativamente extensa» del tales conceptos, que, por otra parte, puede reutilizarse en nuevos

documentos que se realicen sobre una misma temática. Es decir, aunque la percepción del lector pueda ser otra, para el autor, desde un punto de vista instrumental, los glosarios tienen más similitudes con las bases de datos de bibliografía que con los índices terminológicos que MAKEINDEX gestiona.

José Luis Díaz y Javier Bezos [16] han desarrollado el paquete *gloss* con el objetivo de crear glosarios utilizando BIBTEX en lugar de MAKEINDEX, lo que, en nuestra opinión, es la vía adecuada para abordar el tema.

La herramienta *gloss* consta esencialmente de: un paquete para la comunicación con LATEX, de nombre *gloss.sty* y un fichero de estilo para BIBTEX, de nombre *glsplain bst*. El fichero de estilo ha sido desarrollado para gestionar bases de datos con el siguiente formato específico de registros y campos:

```
@GD{Etiqueta
word={Concepto},
definition={DescripciónDelConcepto}
}
```

Es decir, únicamente hay un tipo de registro, @GD, con los siguientes campos obligatorios:

word Donde el *Concepto* debe aparecer tal y como figura en el texto fuente. El estilo *glsplain* será el encargado de convertir a mayúscula la primera letra de *Concepto*. En el glosario que se genere, las entradas con la misma inicial aparecerán agrupadas, encabezadas por dicha letra y ordenadas alfabéticamente.

definition El argumento *DescripciónDelConcepto* puede ser tan largo como permita la longitud máxima soportada en los campos por BIBTEX. Puede contener varios párrafos (realizados con \par) y cualquier construcción de LATEX.

Los comandos básicos para la generación de un glosario, supuesto que se dispone de una base de datos BIB con el formato antes descrito, son los siguientes:

\makegloss	\gloss [<i>Opciones</i>] { <i>Etiqueta</i> }	\printgloss{ <i>BaseDeDatos</i> }
------------	--	-----------------------------------

El comando \makegloss debe colocarse en el preámbulo, junto con el correspondiente paquete *gloss*. Es el encargado de crear el fichero auxiliar, *MiDoc.tex.gls.aux*, en el que se irá anotando cada *Etiqueta* existente en el documento fuente que hayan sido seleccionada para ese fin mediante el comando \gloss. El comando \gloss es análogo al comando \cite. Finalmente \printgloss es un comando similar \bibliography: se ocupa de imprimir el glosario en el documento, utilizando para ello el fichero *MiDoc.tex.gls.bbl* generado por BIBTEX a partir de *MiDoc.tex.gls.aux*, el estilo *glsplain bst* y la base de datos declarada en el argumento *BaseDeDatos*.

Como se observa el flujo de funcionamiento es paralelo al que se emplea para construir la bibliografía de forma automática.

El paquete *gloss* implementa un entorno *thegloss* cuyo título está almacenado en el comando

\glossname

Sin entrar en los detalles, señalaremos también que los registros @GR admiten como campos opcionales los siguientes `short`, `short-word`, `group` y `heading`, y que algunos de estos campos están relacionados con *Opciones* del comando `\gloss` para controlar el formato con que aparece el *Concepto* en el texto. Pueden generarse varios glosarios y es posible imprimir todos los registros usando el comando `\gloss[nocite]{*}`. El lector interesado en este paquete encontrará en la documentación que lo acompaña información complementaria a esta breve presentación.

PARA SABER MÁS

- ▶  Algunos autores necesitan incluir al inicio o al final de un capítulo un breve índice general con los contenidos del mismo. El paquete `minitoc` proporciona herramientas para realizar esa tarea. En el archivo `MiniToC.pdf` se describe el paquete así como el uso de `mtcoff` que lo desactiva.
- ▶  Un aspecto que afecta globalmente a un documento es el de las referencias cruzadas. En la lección 8 ya tratamos este tema; sin embargo, no nos hemos ocupado de él en este capítulo. Existen algunos paquetes que aportan mayor versatilidad a las referencias cruzadas; todos ellos están descritos en el archivo `MasReferencias.pdf`. Allí encontrará, por ejemplo, cómo hacer referencias a otros documentos con el paquete `xr`, cómo incluir en las referencias a las unidades de estructura el título de las mismas (paquete `titleref`) y cómo automatizar la inclusión de textos del tipo «en la página siguiente» o «en el capítulo anterior», a la hora de realizar algunas referencias (paquete `variorref`).

Construcciones

Adiferencia de los dos capítulos anteriores, en que los temas tratados eran de carácter básico general y global, en este tercero nos ocupamos de completar la información sobre construcciones concretas: color, listas, tablas, objetos flotantes y cajas. A decir verdad, estos últimos elementos, las cajas, son tan generales que bien podríamos haberlos ubicado en el capítulo 1. Hemos optado por esta ubicación porque la exposición realizada alcanza niveles de complejidad considerables. Si hasta ahora le ha apasionado L^AT_EX, llegará cerca de la «locura» cuando se sumerja en el mundo de las cajas de T_EX (sección 3.5).

Son tantas las herramientas y paquetes disponibles para las construcciones tratadas en este capítulo que nos hemos visto obligados a realizar una selección, siempre dolorosa. Consulte los apartados PARA SABER MÁS al final de las secciones 3.3 y 3.4, así como la del final del capítulo, para conocer otras posibilidades y dónde aprenderlas.

3.1. Nuevas posibilidades con el paquete color

En el apartado I.5.5 introdujimos las opciones básicas del paquete `color` y dos comandos básicos para escribir textos en color. En esta sección se describen los distintos modelos de color y la forma de definir nuevos colores, una posibilidad interesante aunque, probablemente, de uso esporádico, dados los ya numerosos colores que se nos ofrecen con la opción `usenames` (recuerde los 68 colores predefinidos, mostrados en el cuadro I.5.1). La sección finaliza presentando una sintaxis extendida para los comandos `\color` y `\textcolor` y con las herramientas del mismo paquete para colorear cajas y páginas, las más interesantes, sin duda, de la sección.

Ya conocemos los dos controladores fundamentales de color, `dvips` y `pdftex`, véase el apartado I.5.5; el cuadro 3.1 contiene la lista completa de las opciones de color referidas al controlador, que coincide con la de los paquetes `graphicx` e `hyperref`.

El paquete `color` proporciona la opción `monochrome`, además de las ya estudiadas en el apartado I.5.5. Cuando se incluye esta opción todos los comandos de color son desactivados, de forma que no se genera nada en color, pero no es necesario eliminar los comandos de color, no producen errores. Es una opción muy útil si estamos trabajando con un visor o impresor que no soporta el color o para versiones en borrador.

dvipdf	dvipdfm	dvips	dvipsone	dviwin	dviwindo
emtex	oztex	pctex32	pctexhp	pctexps	pctexwin
pdftex	tcidvi	textures	truetex	xdvi	

Cuadro 3.1: Controladores disponibles como opción para los paquetes color, graphicx e hyperref (véase el apartado I.5.5).

Además de éstos hay otros controladores disponibles, pero no se instalan por defecto ya que se les considera anticuados o poco fiables; éstos son: dvi2ps, dvialw, dvilaser, dvitops, ln, psprint y pubps. Para instalar cualquiera de ellos consulte [10, pág.2]

3.1.1. Modelos y definición de colores

El soporte del color se realiza partiendo de la idea de un *modelo de color*. Los modelos de color soportados varían de un controlador a otro, pero básicamente son:

`rgb` modelo *Rojo-Verde-Azul* (Red-Green-Blue), usado fundamentalmente en monitores y televisión. Se especifica mediante tres números entre 0 y 1, cada uno de los cuales indica, respectivamente, la cantidad de rojo, verde y azul que posee el color.

`cmyk` modelo *Azul-Magenta-Amarillo-Negro* (Cyan-Magenta-Yellow-Black), usado por la industria impresora. De nuevo se especifica mediante cuatro números entre 0 y 1.

`gray` modelo de *Tones Grises*. Cada tono está determinado por su proximidad al blanco o al negro. Se especifica mediante un número entre 0 (negro) y 1 (blanco).

`named` modelo en el que los colores son referidos con un nombre (habitualmente en inglés), por ejemplo, `white`, `green` o `brown`. Estos nombres deben ser conocidos por el controlador, aunque también podemos asignar nombres a los nuevos colores que definamos. Este es el modelo que se maneja cuando se recurre a las opciones `usenames` y `dvipsnames` para incluir los 68 colores predefinidos por James Hafner (véase la página 46).

Cualquier controlador debe tener predefinidos los colores `black` (negro), `white` (blanco), `red` (rojo), `green` (verde), `blue` (azul), `cyan` (azul celeste), `magenta` (magenta) y `yellow` (amarillo). Pero el usuario puede definir cualquier color mediante el comando

```
\definecolor{NombreColor}{Modelo}{Especificación}
```

NombreColor define el nombre con el que nos vamos a referir al nuevo color.

Modelo indica el modelo de color respecto al cual está definido el último argumento.

Especificación es la definición del color *NombreColor* expresada en el *Modelo* fijado. Los números necesarios en esta *Especificación* deben ser separados por comas.

Por ejemplo, podríamos definir algunos nuevos colores como sigue:

```
\definecolor{AzulClaro}{rgb}{0.8,0.85,1}
\definecolor{GrisOscuro}{gray}{0.75}
\definecolor{AmarilloVerdoso}{cmyk}{0.15,0,0.69,0}
```

3.1.2. Sintaxis extendida para texto en color

En el apartado I.5.5 ya introducimos los comandos `\color` y `\textcolor`, cuya sintaxis podemos recordar:

`\color{NombreColor}`

`\textcolor{NombreColor}{Objeto}`

El primero es una declaración que establece el color *NombreColor* como el color por defecto a partir de ese momento, mientras que el segundo escribe *Objeto* en el color *NombreColor*. Conviene advertir que todas las actuaciones descritas se refieren al color de los objetos L^AT_EX y no afectan a objetos externos como gráficos y similares. En ambos casos el color *NombreColor* debe estar ya definido. Estos dos comandos admiten una sintaxis extendida que posee la ventaja de permitir incluir la especificación del color en el comando; esto evita tener que definir un color nuevo y asignarle un nombre que luego será utilizado en el argumento de estos comandos, demasiadas tareas, y consumo de memoria, si el uso de dicho color va a ser muy puntual. La citada sintaxis extendida es:

`\color[Modelo]{Especificación}`

`\textcolor[Modelo]{Especificación}{Texto}`

donde *Modelo* y *Especificación* tienen el mismo significado que en `\definecolor`.

3.1.3. Cajas y páginas en color

Los siguientes comandos construyen cajas con fondo en color. Existen dos versiones de cada comando: la primera se utiliza si los colores que vamos a utilizar han sido previamente definidos, mientras que la segunda está indicada para utilizar colores no definidos.

`\colorbox{NombreColor}{Objeto}`

`\colorbox[Modelo]{Especificación}{Objeto}`

Este comando es análogo a `\mbox`, crea una caja con el fondo en color *NombreColor* y en su interior coloca el *Objeto*. La segunda versión, extendida, utiliza el color definido por los valores de *Especificación*, en el modelo de color dado por el argumento *Modelo*.

El siguiente comando, en sus dos versiones, es el análogo a `\fbox` para cajas en color:

`\fcolorbox{NombreColorBorde}{NombreColor}{Objeto}`

`\fcolorbox[ModeloBorde]{EspecificaciónBorde}[Modelo]{Especificación}{Objeto}`

Construyen una caja con el fondo en color *NombreColor*, colocando en su interior el *Objeto*, pero además le añade un marco a la caja de color *NombreColorBorde*. Estos comandos utilizan los parámetros `\fboxrule` y `\fboxsep` asociados al comando `\fbox` (véase la página 148).

El color de fondo de las páginas puede ser definido mediante el comando

`\pagecolor{NombreColor}`

`\pagecolor[Modelo]{Especificación}`

donde los argumentos *NombreColor*, *Modelo* y *Especificación* son los mismos que en los comandos `\color` y `\textcolor`. Es una declaración global que sirve para la página actual y para todas las siguientes, hasta que se cambie con un nuevo comando `\pagecolor` o se desactive con `\pagecolor{white}`.

EJEMPLO 3.1

```
\usepackage[dvipsnames,usenames]{color}...
\fcolorbox{Blue}{CarnationPink}%
{\textcolor{White}{\TeX}} no fue
diseñado teniendo el \colorbox{Melon}{color}
en mente, por lo que la generación de
\textcolor{OliveGreen}{colores} ....
\colorbox{Gray}{\textcolor{Goldenrod}{opciones}}
descritas ...estén
\colorbox[cmyk]{0,0,0,0.2}{disponibles}.
```

Coloreando cajas

\TeX no fue diseñado teniendo el **color** en mente, por lo que la generación de colores en un documento depende totalmente del controlador. Por esta razón, dependiendo del controlador utilizado, puede que algunas de las **opciones** descritas anteriormente no estén **disponibles**.

3.2. Personalización de listas

CONOCEMOS ya los tres entornos básicos para realizar listas de ítems o apartados: se trata de los entornos `enumerate`, `itemize` y `description`, estudiados, todos ellos, en la lección 11. Además sabemos que algunas opciones de idioma del paquete `babel` alteran el diseño de estas listas a fin de adaptarlo a las convenciones tipográficas del idioma en cuestión (véase la página 174). En esta sección mostraremos cómo realizar modificaciones sobre cualquiera de estos diseños, así como la manera de definir nuevos entornos de este tipo. Los conocimientos para realizar ambas tareas son prácticamente los mismos: se trata de comprender cómo se construyen internamente las listas.

Cualquier modificación de estos entornos debe proceder por etapas, cada una de las cuales, en todo caso opcional, afecta a uno de los elementos de la lista. Las citadas etapas afectarán a los siguientes aspectos:

- la numeración de los ítems (en las listas numeradas): los contadores que la controlan y su representación, así como las referencias cruzadas a dichos ítems;
- la etiqueta de los ítems: diseño de las viñetas en el caso de los entornos `itemize`, de la etiqueta descriptiva en el caso de los `description` y de la numeración de los ítems en el caso de los `enumerate`;
- diseño general de cada ítem y de la lista global: sangrados de cada uno de los niveles, espacio después de cada etiqueta, espacio entre párrafos dentro de un ítem, espacio antes y después del entorno, etc.

Trataremos en primer lugar lo concerniente a las dos primeras etapas. Después introduciremos el entorno `list`, un entorno genérico de tipo lista, que nos permitirá definir nuestros propios entornos de este tipo, para lo cual necesitaremos conocer los parámetros relacionados con la tercera etapa, que será, por tanto, cubierta en ese momento.

3.2.1. Contadores, viñetas y etiquetas

L**A****T****E****X** y algunas opciones de `babel` numeran los ítems de acuerdo con los criterios que tienen definidos. No obstante, sabemos que las etiquetas pueden introducirse explícitamente en los en-

	Nivel 1	Nivel 2	Nivel 3	Nivel 4
<i>contador</i>	<code>enumi</code>	<code>enumii</code>	<code>enumiii</code>	<code>enumiv</code>
<i>representación</i> <i>definición (defecto)</i>	<code>\theenumi</code>	<code>\theenumii</code>	<code>\theenumiii</code>	<code>\theenumiv</code>
<i>etiqueta</i> <i>definición (defecto)</i> <i>ejemplo</i>	<code>\labelenumi</code> <code>\theenumi.</code> 1.	<code>\labelenumii</code> <code>(\theenumi)</code> (a)	<code>\labelenumiii</code> <code>\theenumii.</code> i.	<code>\labelenumiv</code> <code>\theenumiv.</code> A.
<i>prefijo en referencias</i> <i>definición (defecto)</i> <i>ejemplo</i>	<code>\p@enumi</code> <code>{}</code> 1	<code>\p@enumii</code> <code>\theenumi</code> 1a	<code>\p@enumiii</code> <code>\theenumi(\theenumii)</code> 1(a)i	<code>\p@enumiv</code> <code>\p@enumii\theenumiii</code> 1(a)iA

Cuadro 3.2: Contadores y etiquetas para el entorno enumerate

tornos `enumerate` e `itemize` de forma análoga a la utilizada en el entorno `description` y así modificar el formato de la numeración de los ítems. Tal modo de proceder no resulta satisfactorio porque no es automático, y, por ejemplo, introducir un nuevo ítem en la lista supondría tener que rehacer las etiquetas de los ítems ya introducidos. Este cambio manual debe ser limitado a situaciones muy específicas, en las que se desea una o más etiquetas distintas al resto.

Es posible que deseemos hacer cambios en el diseño de las etiquetas para un entorno concreto o para todos los de un tipo. Si se introducen las redefiniciones oportunas fuera de los entornos, todas las instancias del entorno modificado, posteriores a la modificación, se verán afectadas (siempre y cuando, naturalmente, la redefinición no se haya encerrado en un grupo). Si sólo deseamos cambiar una instancia concreta debemos realizar las redefiniciones inmediatamente después del `\begin{Entorno}` en cuestión.

A continuación se describe la forma de realizar modificaciones en cada uno de los entornos `enumerate`, `itemize` y `description` por separado. Observe, en lo que sigue, que los comandos mediante los que se especifica lo que se imprimirá en cada etiqueta contienen el término `label` en su nombre.

Listas numeradas

Cada `enumerate` posee un *nivel* que indica el grado de anidación: un `enumerate` posee nivel 1, un `enumerate` dentro del anterior posee nivel 2, etc. El contador de los ítems es distinto para cada nivel: al nivel uno le corresponde el contador `enumi`, al nivel dos le corresponde `enumii`, y así se tienen también `enumiii` y `enumiv`. Lógicamente, cada uno de ellos se incrementa con un `\item` del nivel correspondiente. Como cualquier otro contador, cada uno de ellos posee una representación `\theenumi`, etc., cuya definición por defecto en L^AT_EX (babel y sus opciones pueden cambiarla) se muestra en el cuadro 3.2.

Una cosa muy distinta a la representación de estos contadores es la etiqueta de los ítems. Normalmente la etiqueta contendrá a dicha representación (de no ser así no se trataría de una lista numerada), pero puede contener otros elementos. A cada nivel se asocia un comando que contiene lo que se imprimirá como etiqueta; se tienen así

<code>\labelenumi</code>	<code>\labelenumii</code>	<code>\labelenumiii</code>	<code>\labelenumiv</code>
--------------------------	---------------------------	----------------------------	---------------------------

En el cuadro 3.2 se muestra su definición por defecto. Por ejemplo, `\labelenumii` se define como `(\theenumii)`, lo que, junto a la representación del contador `enumii`, nos da etiquetas numeradas con letras minúsculas y entre paréntesis: (a), (b)... Podríamos cambiar esta definición para poner letras itálicas y un solo paréntesis recto, para lo que bastaría con:

```
\renewcommand*\labelenumii{\textit{(\theenumii)}}
```

En el ejemplo siguiente puede observar otra redefinición que procede incluyendo la representación `\theenumi` en `\labelenumii`.

EJEMPLO 3.2

La familia de los teclados incluye:
`\begin{enumerate}`
`\renewcommand*\labelenumii{\theenumi.\theenumii.}`
`\item Órganos`
`\item Pianos`
`\begin{enumerate}`
`\item Gran cola`
`\item Media cola o colín`
`\item Vertical\label{Vertical}`
`\end{enumerate}`
`\item Clavicordios`
`\end{enumerate}`
 Debido a su reducido tamaño `\ref{Vertical}`
 es la opción doméstica más extendida

La familia de los teclados incluye:
 1. Órganos
 2. Pianos
 2.a. Gran cola
 2.b. Media cola o colín
 2.c. Vertical
 3. Clavicordios
 Debido a su reducido tamaño 2c es la opción doméstica más extendida

Redefinición de las etiquetas en una lista numerada; la itálica en la representación de `enumii` es introducida por la opción `spanish` de `babel`

Observe en el ejemplo que la referencia al tercer ítem del segundo nivel resulta incorrecta; debería aparecer como ‘2.c’ y no como ‘2c’. La cuestión de las referencias cruzadas a los ítems de las listas se complica ligeramente. L^AT_EX no trata estas referencias como otras: usualmente una referencia a un elemento numerado, generada mediante `\ref`, proporciona la representación del contador de dicho elemento. Si fuera así en este caso que nos ocupa, el comando `\ref{Vertical}` debería haber escrito solamente ‘c’ y, como se ve, no es el caso.

¿Cómo realizar entonces referencias cruzadas correctamente? La solución reside en los siguientes comandos con @:

<code>\p@enumi</code>	<code>\p@enumii</code>	<code>\p@enumii</code>	<code>\p@enumiv</code>
-----------------------	------------------------	------------------------	------------------------

El valor de estos comandos actuará, cuando se utilice `\ref`, como prefijo de la representación del contador referido. En el cuadro 3.2 se muestra la definición que por defecto tienen estos comandos. Se observa que `\p@enumii`, el prefijo para `\theenumii` en el momento de referirse a éste, está definido como `\theenumi`, sin punto posterior, lo que explica el resultado ‘2c’ del ejemplo 3.2.

La complicación no ha terminado. En efecto, después de lo anterior, podemos pensar que basta redefinir los comandos de la familia `\p@enum...` adecuados, y así es, pero para redefinirlos tendremos que prestar atención a la inclusión de los comandos `\makeatletter` y `\makeatother`, ya que los comandos que pretendemos redefinir llevan el carácter @ en su nombre (véase la página 217). El ejemplo 3.3 muestra cómo llevar a cabo estas redefiniciones y cómo resolver el

problema de las referencias cruzadas a los ítems de una lista numerada. Todavía deberíamos comentar algunas cosas más sobre estas referencias.

- En primer lugar, refiriéndonos al ejemplo 3.3, conviene observar que es posible recurrir a otra solución al problema de las referencias. Podríamos haber redefinido la representación de `enumii`, mediante

```
\renewcommand*\theenumii{\thenumi.\arabic{enumii}}
\renewcommand*\labelenumii{\theenumii.}
```

pero, en ese caso, `\ref{SaxoAlto}` habría proporcionado el texto ‘22.2’, puesto que la definición por defecto de `\p@enumii` seguiría activa. Entonces la redefinición anterior debería ser acompañada de:

```
\makeatletter\renewcommand*\p@enumii{} \makeatother
```

El resultado será bueno con los códigos anteriores, pero globalmente no es equivalente a la solución dada en el ejemplo: en ésta el comando `\theenumii`, utilizado directamente, sólo producirá un número, mientras que con el código anterior produciría dos números separados por un punto. ¿Cuál de las dos soluciones es mejor? Todo depende de sus necesidades, de si requiere utilizar `\theenumii` explícitamente y en qué forma.

- En segundo lugar, observemos que ninguna de las partículas adicionales que pueden introducirse en la definición de los `\labelenum...` aparece al hacer las referencias cruzadas; es el caso del ejemplo 3.3 en el que la etiqueta termina con un punto que no se refleja en la referencia. En principio no disponemos de herramienta alguna para conseguir evitar este hecho, aunque siempre podemos añadir manualmente estas partículas. Sin embargo, conviene observar que los expertos en tipografía recomiendan precisamente que esto no se haga; por ejemplo, si las etiquetas en los ítems son de la forma *a), b), c)*, etc., nos recomiendan que las referencias se hagan en la forma *a, b, c...* (véase, a este respecto, el término «apartado» en [60]).

EJEMPLO 3.3

```
La familia de los instrumentos de viento ...
\begin{enumerate}
\renewcommand*\theenumii{\arabic{enumii}}
\renewcommand*\labelenumii{\theenumi.\theenumii.}
\makeatletter
\renewcommand*\p@enumii{\theenumii.}
\makeatother
\item Oboes
\item Saxos
\begin{enumerate}
\item Tenor
\item Alto
\item Soprano
\item Barítono
\end{enumerate}
\item Clarinetes
\end{enumerate}
El más popular es \ref{SaxoAlto}, por su ...
```

La familia de los instrumentos de viento incluye:

1. Oboes
2. Saxos
 - 2.1. Tenor
 - 2.2. Alto
 - 2.3. Soprano
 - 2.4. Barítono
3. Clarinetes

El más popular es 2.2, por su versatilidad y su expresividad, especialmente en grupos de música ligera, jazz y las llamadas «big bands»

Consiguiendo referencias correctas a los ítems de una lista numerada

	Nivel 1	Nivel 2	Nivel 3	Nivel 4
comando definición (defecto) resultado	\labelitemi \textbullet •	\labelitemii \bfseries\textradash —	\labelitemiii \textasteriskcentered *	\labelitemiv \textperiodcentered .

Cuadro 3.3: Etiquetas para el entorno itemize

Listas con viñetas

Los entornos `itemize` son más sencillos que los `enumerate` a la hora de modificar el aspecto de la etiqueta que se imprime para cada ítem. Puesto que no manejan contadores no tenemos la complicación de su representación, ni tampoco la de las referencias, ya que los ítems de estas listas no son referenciables. La etiqueta de los ítem es construida por un comando dependiente del nivel de anidamiento de la lista; estos comandos son

\labelitemi	\labelitemii	\labelitemiii	\labelitemiv
-------------	--------------	---------------	--------------

Para modificar las etiquetas basta redefinir el comando adecuado de entre los anteriores. El cuadro 3.3 muestra las definiciones por defecto de dichos comandos (recuerde que algunos idiomas de babel pueden cambiar estos valores por defecto).

EJEMPLO 3.4

Entre las plantas más populares hay que incluir:

```
\begin{itemize}
\renewcommand*\labelitemi{\clubsuit}
\item Aralia
\item Camedorea
\item[\textbullet] Plantas crasas
\end{itemize}
```

Entre las plantas más populares hay que incluir:

- ♣ Aralia
- ♣ Camedorea
- Plantas crasas

Listas descriptivas

Para este tercer tipo de listas, las correspondientes al entorno `description`, la situación es en parte más sencilla y en parte más complicada que en el caso anterior. Las etiquetas se construyen en términos del argumento optativo del comando `\item` que inicia cada uno de los apartados de estas listas. No es de extrañar, entonces, que el comando responsable de imprimir la etiqueta espere un argumento que, internamente, tomará el valor del citado argumento de `\item`. Este comando que imprime la etiqueta es

\descriptionlabel{Objeto}

Su definición por defecto en las clases estándar es:

```
\newcommand*{\descriptionlabel}[1]{\hspace{\labelsep}\normalfont\bfseries #1}
```

Esta definición merece varios comentarios ya que opera siguiendo una filosofía ligeramente distinta a la de las otras listas. En primer lugar, el comando inserta espacios, algo que no hacen `\labelenumi...` ni `\labelitemi...` (la longitud `\labelsep` se introduce en el apartado 3.2.2).

En segundo lugar, si deseamos cambiar esta definición debemos hacerlo antes de entrar en el entorno `description`, ya que de hacerlo en su interior la redefinición no tendrá efecto. La razón de este cambio será definitivamente aclarada en el apartado 3.2.2, cuando conozcamos el comando `\makelabel`, identificado a `\descriptionlabel` al leer el comando `\begin{description}`, y que es el responsable último de imprimir las etiquetas. Así pues, es posible cambiar la forma de las etiquetas dentro del entorno redefiniendo `\makelabel` y no `\descriptionlabel`.

3.2.2. El entorno `list`

Las técnicas descritas en el apartado anterior son habitualmente suficientes para conseguir personalizar una lista. En caso contrario es posible utilizar el entorno `list` que permite un control completo de la lista. De hecho, los entornos `enumerate`, `itemize` y `description`, e incluso `verse`, `quote`, `quotation` y `thebibliography` (véanse las lecciones 3 y 15) están desarrollados como casos particulares del entorno `list`. La sintaxis de este entorno es:

```
\begin{list}{EtiquetaDefecto}{Declaraciones}
\item Texto1
\item Texto2
...
\end{list}
```

EtiquetaDefecto Es el texto que aparece cuando se escribe `\item`. Si se deja vacío, no aparece etiqueta alguna. En este entorno el comando `\item` también puede llevar un argumento optativo; en caso de incluirlo, es decir, si se escribe `\item[TextoEtiq]`, lo que aparecería como etiqueta de ese ítem sería exactamente el valor de *TextoEtiq*.

Declaraciones Está destinado a contener comandos para fijar uno o varios de los diferentes parámetros de estructura del entorno `list`. Puede estar vacío, en cuyo caso utilizará los que tiene definidos por defecto.

Cuando se abandona un entorno `list` y se retorna al texto normal (o eventualmente a otro entorno `list`) L^AT_EX salta de línea, pero no comienza un nuevo párrafo, a no ser que el usuario lo introduzca explícitamente. Además, se ignoran las líneas en blanco que se introducen en alguna de las siguientes situaciones:

- entre `\begin{list}` y el primer `\item`;
- entre el comando `\item` y el texto de tal ítem (obsérvese el truco empleado en el ejemplo I.11.2 para conseguir un nuevo párrafo inmediatamente después del comando `\item`);
- entre el último ítem y `\end{list}`.

Los entornos `list` se pueden anidar unos dentro de otros hasta seis niveles.

El valor del argumento *Declaraciones* puede incluir comandos referidos a espacios (horizontales o verticales), al uso de contadores, etc. Listamos a continuación los parámetros relativos a longitudes en los entornos `list`, cuyo significado muestra la figura 3.1.

Todos los valores por defecto de estos parámetros pueden modificarse en el segundo argumento de un entorno `list`, a través de los comandos `\renewcommand`, `\addtolength`, etc. Pero

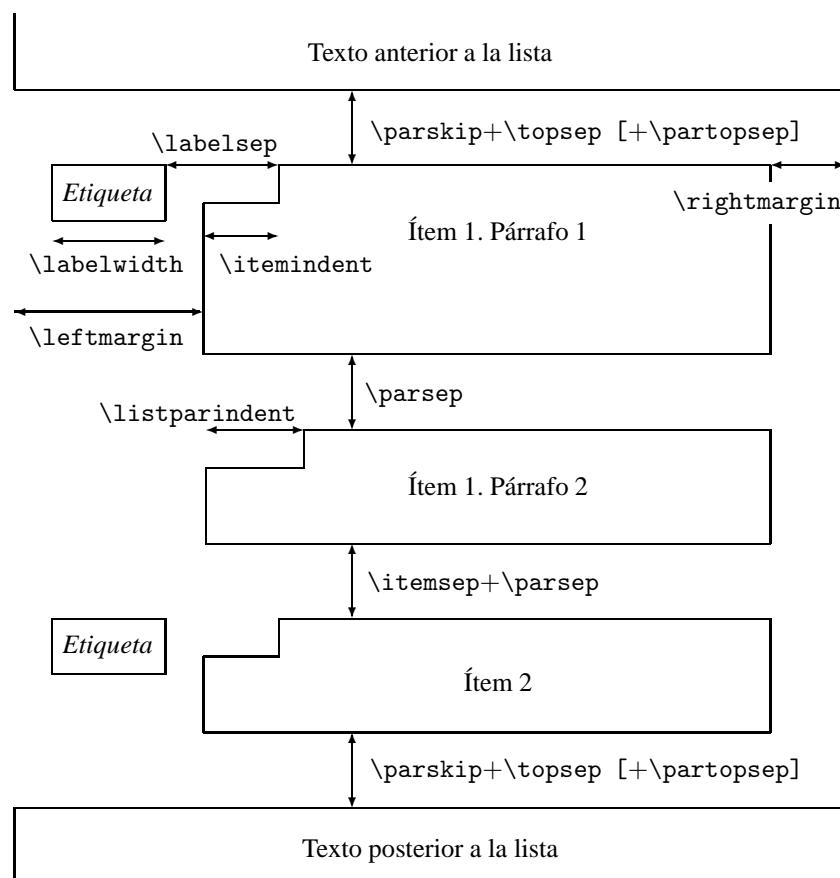


Figura 3.1: Estructura general de una lista del entorno `list`

los entornos `enumerate`, `itemize` y `description` están definidos internamente como entornos `list` particulares y puede ser de interés conocer las posibilidades de modificación en esas listas. Por ello, en la siguiente descripción se agrupan atendiendo a la posibilidad de modificarlos en estos entornos particulares. Así distinguiremos entre los que se pueden modificar dentro del entorno (inmediatamente antes del primer `\item`), los que han de modificarse antes de entrar al entorno y un tercer grupo que sólo pueden ser modificados en el argumento *Declaraciones* de `list`. Dado que la definición de los entornos `enumerate`, `itemize` y `description` maneja un entorno `list`, con su argumento *Declaraciones* ya especificado, algunos parámetros (los del tercer grupo) son inaccesibles para el usuario. La mejor solución, en caso de necesitar dichas modificaciones, sería definir un nuevo entorno (digamos `Mienumerate`), en términos de `list`, que «mimetice» la definición de uno de los entornos estándar (en este caso, sería `enumerate`) pero cambiando el contenido del argumento *Declaraciones* en los aspectos concretos que deseamos modificar. Esta técnica de «copiar» lo definido por clases de documento o paquetes y modificarlo posteriormente es muy útil, aunque requiere una experiencia considerable con L^AT_EX para evitar conflictos.

Observando la figura 3.1 el significado de cada uno de los parámetros parece claro y se diría que todos estos parámetros son independientes. Sin embargo, internamente, algunos están definidos en términos de otros, por lo que modificar uno puede provocar cambios en otros, dando un resultado que, a menudo, sorprende y que a veces parece hacer imposible la comprensión del proceso de formato de los ítems. Por todo ello tratamos de detallar al máximo este proceso, lo que puede resultar en una cierta complicación para el lector.

Modificables dentro del entorno

\itemsep Separación vertical adicional entre ítems consecutivos; generalmente es una longitud positiva o nula, pero también puede ser negativa.

\itemindent Sangrado inicial del párrafo que sigue a `\item`. Generalmente su valor es 0 pt. Puede ser una longitud negativa, como ocurre en el entorno `verse` (véase el ejemplo I.3.6), para conseguir que cuando un verso no cabe en una única línea, la segunda línea del mismo comience con sangría respecto de la línea inmediatamente anterior; también es negativo en el entorno `description`.

\labelsep Espacio horizontal que media entre el comienzo del texto que sigue a `\item` y el final de la caja de la etiqueta. Generalmente su valor es 0,5 em.

\labelwidth Anchura por defecto de la caja que contiene la *Etiqueta* de la lista, ya sea ésta la que tiene fijada la lista o, en su caso, el contenido del argumento optativo del comando `\item`. El valor de `\labelwidth` suele ser igual a la diferencia `\leftmargin-\labelsep`. El entorno `description`, con un valor fijo de 0 pt, es una excepción. En general, si el texto que se imprime en la etiqueta supera la anchura dada por `\labelwidth`, la caja destinada a contenerlo se ensancha, sin producir mensaje de `Overfull`.

Modificables antes del entorno

\partopsep Espacio extra que se añade a `\topsep` antes y/o después de la estructura `list` cuando, en una de estas posiciones, hay un salto de párrafo (`\par` o línea en blanco).

\leftmargin Espacio horizontal de sangrado de la lista respecto del entorno anterior. Es una longitud, normalmente no negativa, que determina la sangría del texto principal de los ítems y su valor depende del nivel de lista de que se trate. En la primera lista toma el valor de la longitud `\leftmargini`, en el segundo nivel de lista toma el valor `\leftmarginii`, y así hasta `\leftmarginvi`, que corresponde al último nivel de anidamiento admitido para `list`. Estas últimas longitudes toman valores constantes que vienen determinados por la clase de documento y sus opciones, y que también se pueden modificar antes de iniciar el entorno.

Algunos de los parámetros descritos hasta aquí son cruciales a la hora de comprender cómo se coloca la etiqueta de los ítems. El proceso puede resumirse en los pasos descritos a continuación.

- En primer lugar se establece un margen izquierdo para toda la lista, que viene dado por `\leftmargin`. Este margen se mide respecto al margen izquierdo natural del texto que precede a la lista, que puede ser texto usual (entonces se aplica `\leftmargini`) o texto en una lista de nivel inmediatamente superior a la actual (y la longitud es `\leftmarginii...`).

- A partir de la posición a izquierda en este margen izquierdo se mueve una longitud igual a `\itemindent`.
- Desde la nueva posición se desplaza hacia la izquierda el espacio dado por `\labelsep` y luego otra vez a la izquierda el dado por `\labelwidth`.
- Llegados a esa posición se imprime la etiqueta, con las precauciones sobre la caja especificadas en la descripción de `\labelwidth`.

En la comprensión de este proceso conviene tener en cuenta algunas observaciones adicionales. En primer lugar, cuando decimos que el desplazamiento es hacia la izquierda estamos suponiendo que las longitudes dadas son positivas; en caso contrario el efecto será el de desplazarse a la derecha. En segundo lugar, puesto que el responsable último de imprimir la etiqueta es el comando `\makelabel`, la posición final de la etiqueta depende de la definición de dicho comando (que cambia de unos entornos a otros); el proceso anterior sólo nos indica la posición a partir de la cual comenzará a actuar `\makelabel`. En particular el control de este comando es fundamental para evitar, por ejemplo, que las etiquetas muy anchas sobresalgan del margen izquierdo de la mancha de texto. Este particular sea analiza con más detalle un poco más adelante.

Modificables en el argumento *Declaraciones* del entorno `list`

- `\parsep` Espacio de separación vertical adicional dentro de un mismo ítem cuando comienza un nuevo párrafo.
- `\topsep` Espacio de separación vertical entre el primer ítem y el texto que le precede, y entre el último ítem y el párrafo que le sigue.
- `\rightmargin` Similar a `\leftmargin` pero referido al margen derecho. Generalmente su valor es 0 pt. No existen las longitudes `\rightmargini`, etc.; es decir, el margen derecho es el mismo para todos los niveles de las listas anidadas.
- `\listparindent` Sangrado adicional al comienzo de cada nuevo párrafo dentro de un mismo ítem (se excluye el que comienza inmediatamente después de `\item`). Generalmente su valor es 0 pt. Puede ser una longitud negativa, como en el entorno `verse`.

`LATEX` proporciona el entorno

```
\begin{trivlist}
\item[Etiqueta1] Texto1
\item[Etiqueta2] Texto2
...
\end{list}
```

Como su nombre indica (*lista trivial*), es una forma simplificada de lista, donde los argumentos de `list` han desaparecido y la etiqueta por defecto (la que incluye `\item` sin argumento opcional) es vacía. Además esta *lista trivial* especifica varios de los parámetros anteriores: `\parsep` es igual a `\parskip`, `\listparindent` igual a `\parindent`, y `\leftmargin`, `\labelwidth` e `\itemindent` son nulos. Puede ser anidado tantas veces como se deseé.

Este entorno es de utilidad para construcciones que no aparentan ser una lista, normalmente con un único `\item` que no introduce etiqueta. De esta forma lo utiliza `LATEX` para definir interna-

mente entornos tan usuales como `center`, `flushright` y `flushleft`, así como `verbatim` (véase el apartado PARA SABER MÁS del capítulo 1) y los generados por el comando `\newtheorem` (véase la sección 5.12). Concretamente, la definición de `center`, por ejemplo, es equivalente a:

```
\begin{trivlist}\centering
\item Texto (cuerpo del entorno)
\end{trivlist}
```

Control de las etiquetas en el entorno `list`

Los entornos `enumerate` utilizan sus propios contadores para la numeración de los ítems. Para crear una lista numerada con el entorno `list` es necesario, además de hacer intervenir un contador (existente o definido con anterioridad al inicio de la lista) en la *EtiquetaDefecto*, que el valor de éste se incremente cada vez que se introduzca un nuevo ítem. Esto podría realizarse por medio de `\refstepcounter`, pero eso nos obligaría a redefinir el comando `\item`, cuya definición es bastante complicada. El siguiente comando cumple esa función específica de forma sencilla:

```
\usecounter{NombreContador}
```

Este comando, situado en el segundo argumento (*Declaraciones*) de un entorno `list`, inicializa el contador *NombreContador* y lo hace avanzar una unidad cada vez que se introduce un `\item` sin argumento optativo. El contador *NombreContador* debe existir previamente. El comando `\label` para las referencias cruzadas a uno de estos ítems maneja este contador. Naturalmente la introducción de este contador en un entorno `list` debe ser complementada con la especificación del formato de las etiquetas, a través del primer argumento *EtiquetaDefecto* del entorno, que deberá contener, de alguna manera, la representación `\theNombreContador`.

El comando responsable, en última instancia, de imprimir las etiquetas de los ítems es:

```
\makelabel{Etiqueta}
```

Se trata de un comando que será llamado por el comando `\item`, pasándole como argumento el texto de la etiqueta del mismo ítem. El argumento será incluido en una caja cuya anchura es el valor máximo entre la anchura del propio argumento y `\labelwidth`. En el entorno `list` la definición por defecto de `\makelabel` se corresponde con:

```
\hfil #1
```

es decir, para una etiqueta de longitud menor que el valor de `\labelwidth`, pone la etiqueta justificada a la derecha (recuerde que, además, la caja que contiene la etiqueta es separada del comienzo del texto normal del ítem por un blanco de anchura `\labelsep`).

Para comprender cómo se comportan los entornos de listas estándar es conveniente saber cómo está definido este comando para cada uno de ellos.

Ya hemos aprendido que en el entorno `description` la definición es equivalente a

```
\renewcommand*{\makelabel}[1]{\hspace{\labelsep}\normalfont\bfseries #1}
```

En este código podemos observar que la etiqueta es sangrada de la longitud `\labelsep`, comportamiento inhabitual en los otros entornos. Por otro lado, observemos que si la etiqueta (#1 en la definición anterior) es muy ancha, el texto con el que se inicia la primera línea del ítem será des-

plazado hacia la derecha; como consecuencia, el texto de la primera línea, excluida la etiqueta, no se iniciará en el mismo lugar en todos los ítems. Este fenómeno se da siempre en el `description`, puesto que todas las etiquetas son más anchas que `\labelwidth`, cuyo valor es, por defecto, 0 pt.

En los entornos `enumerate` e `itemize` la definición es

```
\renewcommand*{\makelabel}[1]{\hss\llap{#1}}
```

Como consecuencia de esta redefinición, la forma de imprimir la etiqueta en los entornos `itemize` y `enumerate` puede producir resultados inesperados. Por ejemplo, las etiquetas de los ítems pueden sobrepasar el margen izquierdo, cuando se utilizan etiquetas más anchas de las que vienen por defecto en la configuración de las clases estándar de documento, tanto si estas etiquetas se han introducido de forma directa mediante `\item[Etiqueta]`, como si son el resultado de una modificación de los comandos `\labelenumi`, `\labelitemi`... Ello se debe al uso de los comandos `\hss` y `\llap` (véase el apartado 1.5.2) en `\makelabel`, que provocan que la etiqueta se imprima ajustada al borde derecho de la caja, de anchura `\labelwidth`, prevista para acogerla y se extienda hacia la izquierda sin ocupar espacio. Aumentar o reducir en tal caso la longitud `\labelwidth` no resuelve el problema, puesto que el punto en el que se empieza a escribir la etiqueta (hacia la izquierda) es siempre el mismo. Para unas etiquetas dadas, excesivamente largas, una forma de resolver el problema es aumentar la longitud `\itemindent` (o aumentar `\leftmargin`, pero naturalmente esto cambiará el formato de toda la lista). Otra posibilidad es redefinir `\makelabel`, asignándole, por ejemplo, el valor que tiene por defecto en el entorno `list` y, eventualmente, disminuir la longitud `\labelwidth`. Observe que una consecuencia de la definición de `\makelabel` en estos entornos es que el sangrado final de la primera línea de cada ítem es el mismo, exactamente `\itemindent`, lo que supone una ventaja.

En lo que resta de esta sección se construyen algunos ejemplos de listas con el entorno `list`, cambiando los valores por defecto de algunos de los parámetros de este entorno.

EJEMPLO 3.5

```
En este ejemplo, tomado de un libro...
\begin{list}{}{
\renewcommand*{\makelabel}[1]%
{\hfil\textsf{#1}:}
\settowidth{\labelwidth}{El árbitro:}%
\setlength{\leftmargin}{\labelwidth}%
\addtolength{\leftmargin}{\labelsep}%
}%
\item[El autor] Como todos los uruguayos,
    quise ser jugador de fútbol. Yo jugaba...
\item[El árbitro] Desde el principio hasta el
    final de cada partido, sudando a mares,
    está obligado a perseguir la blanca pelota...
\item[Los especialistas] Antes del partido,
    los cronistas formulan sus preguntas...
\end{list}
```

En este ejemplo, tomado de un libro de Eduardo Galeano, el autor de *Las venas abiertas de América Latina*, se ilustra la utilización del comando `\makelabel` para modificar la forma en que la etiqueta aparece impresa.

El autor: Como todos los uruguayos, quise ser jugador de fútbol. Yo jugaba muy bien, era una maravilla, pero sólo de noche, mientras dormía.

El árbitro: Desde el principio hasta el final de cada partido, sudando a mares, está obligado a perseguir la blanca pelota que va y viene entre pies ajenos. Es evidente que le encantaría jugar con ella, pero jamás esa gracia le ha sido otorgada. Cuando la pelota por accidente le golpea el cuerpo, todo el público recuerda a su madre.

Los especialistas: Antes del partido, los cronistas formulan sus preguntas desconcertantes.

Sobre el formato de la etiqueta de los ítems; observe que el valor asignado a `\labelwidth` es la anchura de la etiqueta más ancha

EJEMPLO 3.6

En este ejemplo se indica cómo ...

```
\newcounter{MiEnum}
\begin{list}{\fbox{\itshape\theMiEnum.}}
  \usecounter{MiEnum}
  \item Un globo.
  \item Dos globos.
  \item Tres globos.
  \item Muchos globos...
\end{list}
```

Un entorno list con ítems numerados

EJEMPLO 3.7

Este ejemplo ilustra el modo de fijar la ...

```
\begin{list}{\Pisymbol{pzd}{"2F"}{}}
  \item Primer texto
  \item Segundo texto
  \item Tercer texto
\end{list}
Y en este otro se cambia el margen ...
\begin{list}{\Pisymbol{pzd}{'375}{}}
  {\leftmargin 1.5cm}
  \addtolength{\itemsep}{-2mm}
  \item Primer texto
  \item Segundo texto
  \item Tercer texto
\end{list}
```

Sobre la separación vertical y sangría de los ítems

EJEMPLO 3.8

Las etiquetas son tan anchas ...

```
\renewcommand*\labelenumi{Paso \theenumi---}
\begin{enumerate}
\raggedright
\item Desplazamiento de \verb+\leftmargin+
\item Desplazamiento de \verb+\itemindent+
\item Desplazamiento de $-\$\verb+\labelsep+...
\end{enumerate}
Y, ahora, lo mismo corregido
\begin{enumerate}
\raggedright
\newlength{\Desplaza}
\settowidth{\Desplaza}{Paso}
\addtolength{\itemindent}{\Desplaza}
\item Desplazamiento de \verb+\leftmargin+
\item Desplazamiento de \verb+\itemindent+
\item Desplazamiento de $-\$\verb+\labelsep+...
\end{enumerate}
```

Etiquetas de un enumerate que sobresalen por el margen izquierdo

En este ejemplo se indica cómo utilizar un contador en una lista. El recuadro se ha obtenido con el comando `\fbox`, cuyo funcionamiento está descrito en el apartado I.18.1.

- 1.** Un globo.
- 2.** Dos globos.
- 3.** Tres globos.
- 4.** Muchos globos...

Este ejemplo ilustra el modo de fijar la etiqueta

- Primer texto
- Segundo texto
- Tercer texto

Y en este otro se cambia el margen izquierdo y la separación vertical entre los ítems

- Primer texto
- Segundo texto
- Tercer texto

Las etiquetas son tan anchas que sobrepasan el margen izquierdo de la mancha (éste ha sido simulado con un filete vertical).

Paso 1— Desplazamiento de `\leftmargin`

Paso 2— Desplazamiento de `\itemindent`

Paso 3— Desplazamiento de `-\labelsep` y luego de `-\labelwidth`

Y, ahora, lo mismo corregido

Paso 1— Desplazamiento de `\leftmargin`

Paso 2— Desplazamiento de `\itemindent`

Paso 3— Desplazamiento de `-\labelsep` y luego de `-\labelwidth`

EJEMPLO 3.9

```

Y ahora unas etiquetas realmente fantásicas:
\begin{list}{}{%
\renewcommand*\makelabel[1]{%
{\color{Gray}\hfil\rule{\#1}{3pt}}}
\item[0,5\linewidth] La mitad de ...
\item[\labelsep] Separación ...
\item[\itemindent] El valor de ...
\item[\labelwidth] Esta etiqueta tiene
exactamente una anchura igual a
\verb+\labelwidth+=\the\labelwidth
...
\end{list}

```



Y ahora unas etiquetas realmente fantásicas:

- La mitad de la anchura de esta línea de texto normal
- Separación desde el final de la caja de la etiqueta hasta el inicio del ítem
- El valor de \itemindent es nulo
- Esta etiqueta tiene exactamente una anchura igual a \labelwidth=21.90005pt
- ¿Adivina el ancho de esta etiqueta?

3.3. Control y extensiones de las tablas

El entorno `tabular`, estudiado ampliamente en la lección 14, es el marco adecuado de L^AT_EX para la creación de tablas. En esta sección nos ocupamos, en primer lugar, de exponer los parámetros que controlan la creación de estas tablas con `tabular`; su conocimiento le permitirá adecuar un poco más a sus gustos las tablas que cree. En el resto de la sección nos ocupamos de diversos paquetes, cada uno de los cuales resuelve una carencia del entorno `tabular` básico o simplifica una tarea típica con éste. No se incluyen en esta sección más detalles sobre los entornos flotantes asociados a las tablas, es decir, los cuadros (entorno `table`), ya que serán tratados, junto con el entorno `figure`, en la sección 3.4.

3.3.1. Parámetros en las tablas

Los parámetros listados a continuación son útiles para controlar el aspecto de las tablas. Todos ellos deben ser modificados antes de iniciar el entorno `tabular`; una modificación en el interior del mismo produce, según dónde se sitúe, un error o ningún efecto. El primero es un comando, el resto son longitudes modificables, por tanto, con los comandos descritos en el apartado I.17.2.

\arraystretch Controla la separación vertical entre dos filas consecutivas. Su valor por defecto es 1, lo que viene a ser una altura igual al valor de `\baselineskip` (más exactamente, altura igual a 0,7 veces dicha longitud, y profundidad igual a 0,3 veces la misma). Puede ser redefinido, mediante `\renewcommand`, asignándole un valor numérico. Por ejemplo definiéndolo como el valor 1,5 obtendremos una tabla con columnas un 50 % más altas.

\tabcolsep Es la mitad de la separación horizontal entre dos columnas consecutivas, así como el espacio horizontal a la izquierda de la primera columna y a derecha de la última (salvo que se elimine con el especificador `\empty`). Su valor por defecto suele ser 6 pt.

\arraycolsep Se corresponde con la longitud anterior en el entorno matemático `array`.

\arrayrulewidth Determina el grosor de las rayas horizontales y verticales en un entorno `tabular`; su valor por defecto, como para casi todas las rayas de L^AT_EX, es 0,4 pt. Algunas experiencias nos han mostrado que puede ser redefinido al final del argumento de un

comando `\cline`, si bien no parece que esto pueda tener interés. Las rayas muy gruesas pueden producir resultados sorprendentes (véase la sección 3.3.2).

\doublerulesep Determina la separación entre rayas consecutivas o dobles, tanto horizontales como verticales.

EJEMPLO 3.10

```
\begin{center}{\tabcolsep 10pt
\begin{tabular}{||c|c||}\hline
e & ef & efg & efgh \\\hline
\end{tabular}}\quad
\arrayrulewidth3pt
\renewcommand*\arraystretch{2}
\begin{tabular}{|c|c|}\hline
i & ij & ijk & i j k l \\\hline
\end{tabular}\par\bigskip
\arrayrulewidth 2pt\doublerulesep=2pt
\begin{tabular}{||c|c||}\hline
m & mn & mno & mnop \\\hline
\end{tabular}\quad
\begin{tabular}{|c|c|}\hline
q & qr & qrs & qrst \\\hline
\end{tabular}\end{center}
```

a abc	ab abcd
----------	------------

e efg	ef efgh
ijk	ijkl

m mno	mn mnop
----------	------------

q qrs	qr qrst
----------	------------

Cambios en los parámetros del entorno `tabular`; observe el último caso: la duplicación de `\cline` no produce un doble filete. No se muestra el código de la primera tabla, en la que no se ha modificado ningún parámetro; el resultado de la tercera tabla es bueno gracias al paquete `array` (véase el ejemplo 3.12)

3.3.2. Ampliando las opciones del entorno `tabular`: el paquete `array`

Construir tablas, con el entorno `tabular` estándar, que posean columnas con formatos cambiantes, por ejemplo, negrita en una columna, itálica o matemáticas en otra, etc., resulta tedioso, porque en cada celda de la tabla hay que declarar el formato del texto de ésta (cada celda forma un grupo).

El paquete `array`, desarrollado por Frank Mittelbach y David Carlisle [47], mantiene las opciones clásicas de los entornos `tabular` y `array` acerca del formato de columnas (`l`, `r`, `c` y `p`), así como de los separadores de columna (`|` y `@`), pero introduce nuevos formatos de columna y un mayor control en los separadores. Además permite declarar, en el argumento del entorno `tabular`, actuaciones que se realizarán, para las columnas indicadas, antes de que se impriman las celdas de cada una de esas columnas y después de que se impriman. De esta forma, por ejemplo, podemos conseguir que los textos de una cierta columna estén en itálica, sin necesidad de tener que usar en cada una de las celdas el comando para los tipos itálicos.

El cuadro 3.4 recoge los detalles de los nuevos formatos de columna y la sintaxis para declarar las actuaciones citadas. Cualquiera de los nuevos tipos de especificador puede ser utilizado en el segundo argumento (*FormatoColumnas*, véase el apartado I.14.3) del comando `\multicolumn`. Los valores `>` y `<` son totalmente optativos y no es preciso incluirlos si no se desea establecer declaraciones que actúen al empezar o terminar cada celda de una columna dada.

Argumentos de los entornos estándar tabular y array	
<code>l r c p{Ancho}</code>	Formatos de columna. Mantienen su significado básico, aunque array esté cargado. En particular <code>p{Ancho}</code> equivale a <code>\parbox[t]{Ancho}</code> .
<code> @{Objeto}</code>	Separadores de columna. <code>@</code> mantiene su significado. En cambio <code> </code> se comporta de otro modo cuando array está cargado: inserta un filete vertical pero el espacio entre dos columnas se incrementa en el valor correspondiente a la anchura de dicho filete.
Nuevos argumentos con el paquete array	
<code>m{Ancho}</code>	Nuevo especificador de columna, equivale a <code>\parbox[c]{Ancho}</code> . Define una columna de anchura <code>Ancho</code> y el contenido de esta columna se imprime utilizando como punto de referencia la mitad de la altura de la caja que lo contiene; así cada entrada aparece centrada en sentido vertical.
<code>b{Ancho}</code>	Nuevo especificador de columna, equivale a <code>\parbox[b]{Ancho}</code> . Define una columna de anchura <code>Ancho</code> y el contenido de esta columna se imprime utilizando como punto de referencia la esquina inferior izquierda de la caja que lo contiene.
<code>!{Objeto}</code>	Nuevo separador de columnas, inserta <code>Objeto</code> entre dos columnas. Se diferencia de <code>@{Objeto}</code> en que no se suprime el espacio normal entre columnas.
<code>>{Decl}</code>	Puede usarse antes de los especificadores <code>l, r, c, p, m, b</code> . El efecto producido es el que correspondería a introducir <code>Decl</code> delante de cada una de las celdillas de la columna. Se utiliza para definir en una columna de la tabla declaraciones que afectan a todos los elementos de la misma.
<code><{Decl}</code>	Puede usarse después de los especificadores <code>l, r, c, p, m, b</code> . El efecto producido es el que correspondería a introducir <code>Decl</code> detrás de cada una de las celdillas de la columna. Se utiliza en el mismo sentido que <code>></code> .

Cuadro 3.4: Especificadores de columna, separadores y argumentos con el paquete array

Si un determinado formato de columna se va a repetir en diferentes tablas (o simplemente varias veces en una misma tabla) un nivel de comodidad, adicional al que acabamos de indicar, sería introducirlo como un nuevo formato de columna, identificándolo mediante una letra, como ocurre con los tipos usuales `l, c, r, p`, y cuya utilización fuera idéntica a la de éstos. El comando siguiente tiene esa función.

```
\newcolumntype{Car}{>{Declarainicio} z <{Declarafin}}
```

Este comando declara al carácter `Car` como el nuevo formato de columna que, de ese modo, se define; `z` indica la forma en que serán justificadas las columnas del nuevo formato y debe ser un especificador de columna, ya sea estándar o de los nuevos introducidos por el paquete; en caso de ser de tipo «párrafo», es decir, `p, m` o `b`, debe ir seguido del argumento `{Ancho}`. `Declarainicio` y `Declarafin` son las actuaciones, ambas optativas, a realizar antes y después de imprimir cada celda de la columna.

Se definen también en el paquete dos herramientas de interés: la primera sirve para mejorar la presentación de la tabla; la segunda permite recordar los tipos de columnas ya definidos.

\extrarowheight En tablas con filetes horizontales, las letras mayúsculas quedan muy cerca de éstos. La longitud `\extrarowheight` permite añadir una altura adicional a cada línea de la tabla sin modificar la profundidad, mejorando el resultado obtenido por defecto.

\showcols Es un comando que permite enviar al terminal y al fichero log los tipos de columnas activos introducidos con `\newcolumntype` y la descripción de su funcionamiento. Aparecen sólo los tipos activos y no, por ejemplo, los que se hayan creado en el interior de un grupo. Una buena práctica puede ser declarar todos los nuevos tipos en el preámbulo del documento.

EJEMPLO 3.11

```
\newcolumntype{L}{>{\itshape r}}
\begin{tabular}{|l|L{---}|}\hline
&NOMBRE &NAME\\\hline
1 & uno & one \\ 2 & dos & two\\
3 & tres & three\\\hline
\end{tabular}\par\medskip
\setlength{\extrarowheight}{2pt}
\begin{tabular}{|l|L{---}|}\hline
&NOMBRE &NAME\\\hline
1 & uno & one \\ 2 & dos & two\\
3 & tres & three\\\hline
\end{tabular}
```

	<i>NOMBRE</i> — NAME
1	<i>uno</i> — one
2	<i>dos</i> — two
3	<i>tres</i> — three

	<i>NOMBRE</i> —NAME
1	<i>uno</i> —one
2	<i>dos</i> —two
3	<i>tres</i> —three

Un nuevo formato de columna con array y efecto de `\extrarowheight`; observe también la diferencia entre los separadores @ y !

El paquete `array` introduce una mejora adicional en los entornos `tabular`: cuando se utilizan filetes de separación gruesos los puntos de unión de filetes verticales y horizontales resultan muy poco satisfactorios, debido a que los horizontales finalizan en mitad de los verticales. En el ejemplo siguiente pueden ser comparados los resultados con y sin `array`.

EJEMPLO 3.12

```
\usepackage{array}
\begin{document}
\renewcommand*\arraystretch{2}\setlength{\arrayrulewidth}{5pt}
\begin{tabular}{|l|}\hline A \\\hline
\end{tabular}

(nuevo documento)
\renewcommand*\arraystretch{2}\setlength{\arrayrulewidth}{5pt}
\begin{tabular}{|l|}\hline A \\\hline
\end{tabular}
```



Filetes gruesos en tablas con y sin el paquete `array`

3.3.3. Tablas grandes: el paquete `longtable`

Cuando L^AT_EX encuentra que un entorno `tabular` no cabe en el espacio que resta de la página que está componiendo, inicia una nueva página y coloca dicho entorno al comienzo de la misma. Así, incluso aunque la tabla no supere la longitud de la página, los resultados pueden llegar a ser

mediocres en algunas situaciones. La forma en que L^AT_EX permite resolver estas situaciones es insertando la tabla como un objeto flotante (un cuadro; para más detalles véase el apartado I.14.4), que imprimirá la tabla en el sitio en que tenga cabida y llenará con el texto que sigue a la tabla el espacio que resta hasta completar la página.

El paquete `longtable`, desarrollado por David Carlisle [9], permite el tratamiento automatizado de tablas cuya altura supera la longitud de la página. También permite la continuación en la página siguiente de una tabla que no ha sido posible imprimir en su totalidad en la página actual. Proporciona, en este segundo caso, una solución diferente a la de incluir la tabla como flotante.

Este paquete define el entorno `longtable` como una extensión del entorno `tabular`, al que puede reemplazar con ventaja. La estructura del nuevo entorno es idéntica a la del entorno `tabular` en cuanto al tratamiento y formato de las columnas, pero incorpora una especie de preámbulo optativo para determinar las leyendas, que pueden actuar ahora en la cabecera y en el pie de la tabla simultáneamente, en las diferentes páginas a las que se extiende.

Un problema que se plantea en tablas que ocupan varias páginas es que el ancho de una misma columna en páginas diferentes puede ir variando. El paquete dispone de herramientas para conseguir que el ancho de las columnas sea homogéneo en las diferentes páginas. Es necesario compilar el documento varias veces hasta que se consigue este objetivo.

```
\begin{longtable}[Posición]{FormatoColumnas}
\caption[TextoLeyendaÍndice]{TextoLeyenda} \\
PrimeraCabecera\endfirsthead
Cabecera\endhead
Pie\endfoot
ÚltimoPie\endlastfoot
Fila1Columna1 & Fila1Columna2 & ... & Fila1ColumnaN \\
Fila2Columna1 & Fila2Columna2 & ... & Fila2ColumnaN \\
... & ... & ... & ... \\
FilaPColumna1 & FilaPColumna2 & ... & FilaPColumnaN
\end{longtable}
```

El argumento optativo *Posición*, que puede tomar uno de los valores `c`, `l`, `r`, determina si la tabla estará centrada (valor por defecto), alineada a la izquierda o a la derecha, respectivamente, en el nuevo párrafo que inicia el entorno.

Los cuatro comandos que siguen, todos optativos, pueden utilizarse al comenzar el cuerpo del entorno `longtable`, antes de que empiece el texto de las columnas de la tabla, tal y como se muestra en el ejemplo de la página 295. Pueden utilizarse, en este comienzo, comandos `\caption` o `\multicolumn`, así como filetes.

\endfirsthead Marca el final de lo que aparecerá como cabecera de la tabla en la primera página de la misma. El texto *PrimeraCabecera* debe estar formado por una o más filas de la tabla o bien por uno o más comandos `\caption`, en cualquiera de las modalidades expuestas un poco más adelante. La misma restricción se aplica a los textos *Cabecera*, *Pie* y *ÚltimoPie*.

\LTpre	Espacio antes de la tabla.	(\bigskip)
\LTpost	Espacio después de la tabla.	(\bigskip)
\LTcapwidth	El ancho de la \parbox que contiene la leyenda.	(4 in)
\newpage	Introducido al final de una línea acabada en \\ inicia una nueva página.	

Cuadro 3.5: Otros parámetros y comandos del entorno longtable (entre paréntesis figura el valor por defecto)

\endhead Marca el final de lo que aparecerá como cabecera de la tabla en cada página de la tabla (salvo, eventualmente, la primera).

\endfoot Marca el final de lo que aparecerá como pie en cada página de la tabla (salvo, eventualmente, la última).

\endlastfoot Determina el final de lo que aparecerá como pie en la última página de la tabla.

\caption[*TextoLeyendaÍndice*]{*TextoLeyenda*} Si se utiliza se imprimirá una leyenda para la tabla, consistente en el valor de \tablename~\thetable seguido de *TextoLeyenda* (es decir, algo similar a lo que realiza el comando \caption del entorno table, véase el apartado 14.4). Además, *TextoLeyenda* aparecerá como una entrada en el índice de cuadros, a menos que se utilice el argumento opcional *TextoLeyendaÍndice*, en cuyo caso será el contenido de este último el que aparezca en el índice de cuadros.

En caso de utilizar este comando es conveniente, aunque no obligatorio, incluirlo al principio de la tabla, especialmente si ésta ocupa varias páginas. Si no se incluye tampoco se debe incluir el comando \\ que le sigue en la sintaxis antes expuesta. El comando admite algunas variaciones:

\caption*{*TextoLeyenda*} No se escribe nada en el índice de cuadros y se imprime únicamente *TextoLeyenda* en la leyenda, sin el antetítulo \tablename~\thetable correspondiente.

\caption[]{{*LeyendaSecundaria*}} Se puede utilizar para imprimir, en las restantes páginas, una leyenda diferente de la utilizada en la primera página de la tabla. Debe incorporarse como parte de los textos *PrimeraCabecera*, *Cabecera*, *Pie* o *ÚltimoPie*. No produce entrada en el índice de cuadros pero repite la que aparece en la primera página con *LeyendaSecundaria* en lugar de *LeyendaPrincipal*. Alternativamente puede usarse el comando \caption*{*OtraLeyenda*} para que se imprima *OtraLeyenda* sin incluir \tablename~\thetable.

Los comandos \hline y \multicolumn, así como el especificador p y el separador @, operan como en el entorno tabular. También es posible incluir el comando \newpage dentro del entorno longtable como se indica en el cuadro 3.5.

En el paquete están definidos otros parámetros que permiten controlar el espacio anterior y posterior al grupo longtable, el hecho de que la tabla aparezca centrada por defecto, etc. En [9] puede encontrarse más información al respecto.

Construcciones

294

Cuadro 1: Escalafón General de los Señores Jefes, Oficiales, Cabos e Individuos que forman la Brigada Municipal de Zapadores Bomberos de la Ciudad de Murcia (1 de Noviembre de 1886)

NOMBRES	CARGOS	FECHA INGRESO
PLANA MAYOR		
D. Julian Pagan y Ayuso	Director	7 Dicbre 1880
D. Salvador Martinez Meseguer	Comandante	5 Mayo 1885
D. José M ^o Bajajena Sanchez	Jefe Sección	21 Enero 1857
D. José González y González	"	10 Enero 1858
D. José M ^o Godínez Soler	Jefe Bomba	30 Octubre 1876
D. Francisco Ilán Sanchez	"	14 Abril 1879
D. Matano del C. Gonzalez Sanz	"	5 Octubre 1881
D. Eugenio Bruscallos Pérez	"	30 Abril 1886
D. José M ^o de Ayllón y Martínez	"	4 Mayo 1886
D. Emilio Lacarcel López	"	28 Nubre 1889
D. Manuel Costa y Farmas	Aux. Jefe Bomba	15 Julio 1891
D. Diego Hernández Illán	"	15 Julio 1891
D. José Minano Ruiz	"	4 Nubre 1895
D. Antonio Ramírez de Cuero	"	22 Abril 1896
D. José Costa Fernández	Brigada	18 Julio 1865
D. José Martínez Carrasco	Guardia-parque	16 Mayo 1865
D. José Antonio Sánchez Martínez	Furriel	20 Octubre 1875
D. Antonio María Hernández	Corneta	30 Julio 1886
D. José Lorente García	Corneta	1 Julio 1895
SANDAD		
D. Miguel Gutiérrez Baeza	Médico	8 Agosto 1881
D. José M ^o Castillo Tapia	"	5 Octubre 1885
D. José Pino Vivo	Farmacéutico	4 Agosto 1882
D. Francisco García Gualer	Capellán	18 Junio 1883
D. Miguel López Atenza	Practicante	7 Dicbre 1895
D. Joaquín Martín Gómez	Practicante	2 Dicbre 1895
CAJAS DE ESCUADRAS Y DE BOMBA		
1 Antonio Alcazar Sánchez	Cabo de Bomba	15 Septiembre 1859
2 Ramón Giménez Pagan	Escuadra	8 Junio 1864
3 José García Gil	Cabo Zapadores	1 Dicbre 1868
4 Lorenzo Macario Carmelo	"	16 Enero 1875
5 Francisco Beltrán Pons	Bomba	5 Agosto 1875
6 José Godínez Morales	Bomba	10 Agosto 1877
7 José V. Martínez Cárnara	Cabo Escuadra	19 Septiembre 1873
8 Juan Marchion Serrano	"	29 Abril 1879
9 José Atx Gámez	Cabo Bomberos	6 Febrero 1890
10 José Moreno Bergante	"	28 Febrero 1890

Continua en la página siguiente

Cuadro 1: Escalafón General (1 de Noviembre de 1886)

Continuación		INDIVIDUOS
11 Francisco Gárate Bermúdez	Zapador	8 Enero 1859
12 Francisco Pérez Pérez	Bombero	15 Julio 1865
2 Francisco Ropero Martínez	"	10 Enero 1878
3 Rafael Nicolás	Zapador	10 Enero 1878
4 Emilio Martínez Sánchez	"	1 Febrero 1882
5 José Garre Bermúdez	Bombero	28 Septiembre 1882
6 Lope Martín Toral	"	22 Nubre 1883
7 José Alentorn Climent	Zapador	1 Junio 1884
8 Juan Martínez Cáñovas	"	1 Octubre 1887
9 José Ant ^o n Fernández García	"	17 Marzo 1877
10 Juan de la Cruz López Valera	"	15 Marzo 1885
11 Francisco Conesa Lozano	Bombero	10 Junio 1886
12 José Quevedo Esteban	Zapador	22 Enero 1887
13 Antonio Ramírez Roja	Bombero	15 Abril 1887
14 José María Peñuelo Martínez	Zapador	10 Febrero 1888
15 Juan José García Illán	"	15 Octubre 1889
16 Francisco Rodríguez Puerto	"	15 Abril 1891
17 Eufrasio Díaz Cortés	"	15 Abril 1891
18 Domingo Roldenes Marín	"	4 Mayo 1891
19 Francisco Matín Vera	"	27 Octubre 1891
20 Salvador Macario Hernández	Bombero	"
21 Francisco Alemany	Zapador	27 Octubre 1891
22 Juan Orenes Orenes	"	11 Abril 1892
23 Antonio Giménez Andrés	"	27 Marzo 1892
24 Antonio Lillo Hernández	"	27 Marzo 1892
25 Antonio López Nagore	Bombero	29 Marzo 1892
26 Francisco Martínez Ramón	"	30 Mayo 1892
27 Diego Gutiérrez Villa	"	20 Septiembre 1893
28 José María Bosque Leal	Zapador	27 Septiembre 1893
29 Gregorio Martínez López	Bombero	1 Noviembre 1893
30 Mariano Barbera Leal	"	30 Nubre 1893
31 Bartolomé Martínez Martínez	"	18 Febrero 1894
32 Juan Antonio Martínez Ruiz	Zapador	8 Julio 1894
33 Antonio Sánchez Viendo	Bombero	18 Julio 1894
34 Miguel Gómez Alcáñara	"	18 Julio 1894
35 Francisco Escudero Sánchez	"	18 Julio 1894
36 Cristóbal Pérez del Toro	Zapador	14 Dicbre 1894
37 Brigido Turpin Roda	Bombero	14 Dicbre 1894

Continúa en la página siguiente

EJEMPLO 3.13

```
\begin{longtable}{|r||c|c|c|}
\caption[Escalafón General de 1896]{%
  Escalafón General de ... la Ciudad de Murcia
  (1 de Noviembre de 1886) \\%
}
\hline\endfirsthead
\caption[] {Escalafón General(1 de Noviembre de 1896)}\\%
\multicolumn{4}{c}{\footnotesize \textit{Continuación}}\\%
\endhead
\multicolumn{4}{c}{\footnotesize \textit{Continúa en la página siguiente}}\\%
\endfoot
\hline \endlastfoot
\multicolumn{2}{|c|}{\footnotesize \textit{NOMBRES}}\\%
& CARGOS & FECHA INGRESO \\
\hline
\multicolumn{4}{c}{\footnotesize \textit{PLANA MAYOR}} \\
D.&Julian Pagán y Ayuso &Director &7 Dicbre 1880\\%
D.&Salvador Martínez Meseguer &
Comandante &5 Mayo 1855\\ ...
\multicolumn{4}{c}{\footnotesize \textit{SANIDAD}} \\
D.&Miguel Giménez Baeza & Médico &8 Agosto 1881 \\
... \end{longtable}
```

(El resultado de este código se muestra en la página 294)

Código fuente del ejemplo de longtable de la página anterior (este ejemplo fue elaborado por uno de los autores para el libro de Eduardo Carrillo Sánchez «La Brigada de Zapadores Bomberos de la Ciudad de Murcia». Ayuntamiento de Murcia, 1896)

3.3.4. Agrupar filas: el paquete multirow

El comando `\multicolumn` del entorno `tabular` permite insertar textos que se extienden a varias columnas de una misma fila. Sin embargo, en el entorno `tabular` estándar, no existe un comando análogo para insertar textos que se extienden a varias filas, dentro de una columna. En el apartado PARA SABER MÁS de la lección 14 ya señalábamos esta carencia como una de las más llamativas en los procedimientos de construcción de tablas. El paquete `multirow`, desarrollado por Jerry Leichter [38], define el comando `\multirow` que permite cubrir esta ausencia de forma cómoda y que puede ser utilizado tantas veces como se desee dentro de una tabla.

`\multirow{Número}{Ancho}[Ajuste]{Texto}`

Número Es el número de filas a agrupar. El autor debe seguir indicando todas las celdas mediante el carácter `&`, pero debe dejar vacío el contenido de las celdas que son ocupadas por la «fila múltiple».

Ancho Determina la anchura de la columna en la que aparecerá el *Texto*.

Texto Es el texto a incluir. Aparecerá en una caja en modo `\raggedright`, es decir, sin justificar por la derecha. Puede usarse `\backslash` dentro de *Texto* para iniciar nuevas líneas donde se deseé. El texto se imprime centrado en sentido vertical dentro del espacio correspondiente al *Número* de filas agrupadas.

Ajuste Es una longitud que permite ajustes finos de la posición vertical del *Texto*. Éste será desplazado verticalmente un espacio igual al valor de *Ajuste*, hacia arriba si *Ajuste* es positiva o hacia abajo en caso contrario.

El siguiente comando determina el modo en que aparece el *Texto* de \multirow:

```
\multirowsetup
```

Su definición por defecto equivale a

```
\newcommand*{\multirowsetup}{\raggedright}
```

Puede ser redefinido para, por ejemplo, conseguir un texto centrado; basta para ello sustituir \raggedright por \centering.

EJEMPLO 3.14

```
\begin{tabular}{|c|c|}\hline
\multirow{4}{1.5in}{Texto en columna 1}
& C2a \\ & C2b \\ & C2c \\ & C2d \\ \hline
\renewcommand*{\multirowsetup}{\centering}
\multirow{3}{1.5in}%
{Más,\textbackslash\textbackslash texto en columna 1}
& C2a \\ \cline{2-2} & C2b \\ \cline{2-2} & C2c \\ \hline
\end{tabular}
```

Texto en columna 1	C2a C2b C2c C2d
Más, texto en columna 1	C2a C2b C2c

Textos que abarcan varias filas en una tabla; observe que se incluyen los caracteres de cambio de columna &, correspondientes a las celdas de la primera columna, filas dos a cuatro, seis y siete, y que dichas celdas aparecen sin contenido

3.3.5. Tablas con filetes dobles: el paquete `hhline`

En las tablas realizadas con el entorno `tabular` las rayas o filetes se imprimen satisfactoriamente siempre que se trate de rayas simples y del grueso que L^AT_EX utiliza por defecto (para rayas gruesas, el resultado puede no ser satisfactorio sin la utilización del paquete `array`, véase el apartado 3.3.2). En algunos casos la utilización de \hline\hline puede producir resultados adecuados, como el mostrado en el ejemplo I.14.7, pero en general no es ésa la situación, como pone de manifiesto la última de las tablas del ejemplo 3.10. El resultado suele ser malo cuando se cruza un filete doble con otro, sea éste simple o doble.

El paquete `hhline`, desarrollado por David Carlisle [7], define, entre otros, el comando \hhline que produce o bien rayas simples, como las que se obtienen con \hline, o bien dobles, como las que se obtienen con \hline\hline, pero que interaccionan mejor con las rayas verticales, como se muestra en el ejemplo 3.15. La mejora de los resultados supone, eso sí, una sintaxis muy particular, bastante más compleja que la usual, entre otras razones, porque no sigue las pautas de la sintaxis de L^AT_EX.

El comando \hline imprime un filete (caja negra) simple, cuya anchura coincide con la anchura del entorno `tabular` y cuya altura (grosor) es la longitud `\arrayrulewidth`. El comando \hhline imprime también una caja de anchura la del entorno pero que está formada por elementos de uno de los dos tipos siguientes: un blanco o un filete horizontal, simple o doble (uno por

cada columna) y grupos de caracteres que determinan el formato del entrecruzamiento con las verticales del *FormatoColumnas* del entorno tabular. La sintaxis de \hhline es:

\hhline{*ColumnasEIntersecciones*}

El argumento *ColumnasEIntersecciones* es similar al argumento *FormatoColumnas* del entorno tabular, sólo que ahora cada columna y cruce de filetes horizontales y verticales debe ser especificado convenientemente. Esta especificación en *ColumnasEIntersecciones* se realiza por medio de los siguientes elementos:

- = Una raya horizontal doble del ancho de una columna.
- Una raya horizontal simple del ancho de una columna.
- \ Una columna sin raya horizontal.
- | Una raya vertical que corta a una horizontal (simple o doble).
- : Una raya vertical que es partida por una horizontal doble. Detrás o delante debe haber =.
- # Dos rayas verticales que cortan a una horizontal doble.
- t La semiparte superior de una horizontal doble.
- b La semiparte inferior de una horizontal doble.

Además podemos incluir la sentencia

*{*Núm*} {*Argumento*}

que equivale a repetir el *Argumento* tantas veces como indica *Núm*; por ejemplo, *{3}{==#} se expande como ==#==#==#.

Todos los elementos anteriores se usan agrupados, para definir los entrelazamientos con las rayas verticales. He aquí algunas agrupaciones típicas. Entre paréntesis se muestra el resultado que produce cada una de ellas.

- |t: Define la esquina superior izquierda de la intersección de dos filetes dobles, uno vertical y otro horizontal (↖).
- :t| Como el anterior pero referido a la esquina superior derecha (↗).
- |b: Como los anteriores pero referido a la esquina inferior izquierda (↙).
- :b| Similar a los anteriores pero referido a la esquina inferior derecha (↘).
- | : Una raya vertical seguida de otra vertical que se cruza con una horizontal doble (፤).
- : | Una raya vertical, que se ha cruzado con una horizontal doble, seguida de otra vertical (፤).
- :: Un enlace sin cortes entre dos filetes dobles verticales/horizontales (፤).
- || Un filete vertical doble que no es atravesado por los filetes horizontales () .

Cuando se especifica una raya vertical doble (con || o ::) las horizontales producidas con \hhline no la atraviesan. Para obtener un corte de rayas dobles hay que usar #.

Los elementos t y b deben usarse entre dos verticales. Su uso principal es en las esquinas superiores e inferiores, como se han mostrado en los ejemplos anteriores. La combinación |tb| produce las mismas rayas que #, pero es menos eficiente.

Si se utiliza el comando `\hhline` para hacer una raya horizontal simple, su argumento puede contener únicamente `-`, `~`, `|` y expresiones `*`.

El ejemplo siguiente muestra las variadas posibilidades de intersección.

EJEMPLO 3.15

```
\begin{center}\begin{tabular}{||cc||c|c||}
\hhline{|t==:t==:t|} 
a & b & c & d \\ 
1 & 2 & 3 & 4 \\
i & j & k & l \\
m & n & o & p \\
q & r & s & t \\
w & x & y & z \\
\end{tabular}\end{center}
```

a	b	c	d
1	2	3	4
i	j	k	l
m	n	o	p
q	r	s	t
w	x	y	z

Tabla con filetes dobles

3.3.6. Tablas con color: el paquete `colortbl`

La posibilidad de introducir color en los documentos L^AT_EX es una potencialidad siempre de agradecer. Ya sabemos cómo escribir texto en color y cómo dar color al fondo y a los marcos de las cajas. En esta sección presentamos el paquete `colortbl` cuyo objetivo es dar color al fondo de las tablas, ya sea a las columnas, a las filas o a ambas. Usualmente las tablas con color no utilizan filetes de separación, ya que los cambios de color son precisamente los que permiten hacer la separación visualmente clara; a pesar de ello el paquete mantiene la posibilidad de utilizar estos filetes de separación e, incluso, de darles color.

El paquete `colortbl` basa su funcionamiento en los paquetes `color` y `array` que se cargan automáticamente al cargar aquél. Además podemos incluir opciones al cargar `colortbl` que serán «transmitidas» a `color`; deben ser, pues, opciones admitidas por éste, por ejemplo, opciones de controlador (véase el cuadro 3.1 en la página 274). La sintaxis de los comandos de `colortbl` está basada directamente en la posibilidad de declarar nuevos formatos de columna, mediante el carácter `>`, y el comando `\newcolumntype`, proporcionada por `array` (véase la página 290). Por otra parte, y como regla general, el paquete funciona con los paquetes cuya sintaxis sea compatible con la de `array`, como, por ejemplo, `longtable`.

Quizás resulte interesante tener en mente la idea de que `colortbl` procede colocando un «panel» o tapiz de color debajo del contenido de las celdas de una tabla y que éste se coloca verticalmente (por columnas) u horizontalmente (por filas). El comando para colorear columnas es

```
\columncolor [Modelo] {Color} [SepIzq] [SepDer]
```

Los dos primeros argumentos se refieren a la determinación del color que se dará a la columna. El primero, optativo, es el *Modelo* de color y, si se utiliza éste, el segundo argumento debe consistir en la especificación de un color adecuada al *Modelo* seleccionado (véase la sección 3.1). Si no se incluye el argumento *Modelo*, entonces *Color* puede ser un nombre de color admitido por el controlador, por ejemplo, uno de los 68 nombres de `dvipsnames` (véase el apartado I.5.5).

Los argumentos *SepIzq* y *SepDer* son longitudes que indican la distancia, a izquierda y derecha respectivamente, que el panel de color debe exceder del espacio ocupado por el texto de las celdas de la columna (que será el ocupado por la celda más ancha), véase el ejemplo 3.16. Si no se incluye *SepDer* pero sí *SepIzq*, el primero toma el valor del segundo. Si no se incluye ninguno de ellos el valor que se toma para ambos es el valor por defecto, que es `\tabcolsep` o `\arraycolsep`, según estemos en un entorno `tabular` o `array`, respectivamente. Con estos valores por defecto no existirá espacio blanco entre la columna coloreada y las columnas precedente y posterior.

Es fundamental recordar que el comando `\columncolor` sólo puede utilizarse en el argumento *Decl* del especificador de columna `>{Decl}` del paquete `array` o en la definición de un nuevo tipo de columna a través de `\newcolumntype`. También puede incluirse dentro de un comando `\multicolumn`, pero siempre con `>{Decl}` en el argumento de este último.

EJEMPLO 3.16

```
\begin{tabular}{|>{\columncolor[gray]{0.9}}l
               >{\color{White}\columncolor[gray]{0.6}}r|}r|l
  alfa & beta\\ gamma & delta
\end{tabular}\hspace{1cm}
\begin{tabular}{|>{\color{Yellow}\columncolor[gray]{0.7}[0pt]}l
               >{\columncolor{Yellow}[0.5\tabcolsep]}r|}r|l
  alfa & beta\\ gamma & delta
\end{tabular}
```

Columnas de una tabla en color

alfa	beta
gamma	delta

alfa	beta
gamma	delta

Para dar color horizontalmente disponemos del comando:

`\rowcolor [Modelo] {Color} [SepIzq] [SepDer]`

Este comando debe aparecer al comienzo de una fila; de lo contrario, el compilador emitirá un mensaje de error. El significado de los argumentos es el mismo que en `\columncolor`. Si se utilizan `\columncolor` y `\rowcolor` en una tabla existirá al menos una celda afectada por ambos; en tal caso es el color asignado por `\rowcolor` el que prevalece en dicha celda (ya que es el último asignado).

EJEMPLO 3.17

```
\begin{tabular}{|l|r|}\hline
\rowcolor[gray]{.9} alfa & beta\\\hline
\rowcolor[gray]{.7} gamma & delta\hline
\end{tabular}\par\bigskip
\begin{tabular}{|l|r|}\hline
\rowcolor[gray]{.9}[0pt] alfa & beta\\\hline
\rowcolor[gray]{.7}[0pt]\color{White}gamma &
\multicolumn{1}{>{\columncolor{Green}}%
[0pt]}{\color{White}delta}\\\hline
\end{tabular}
```

alfa	beta
gamma	delta

alfa	beta
gamma	delta

Filas de una columna en color

No existe ningún comando para asignar color a una celda en particular, aunque esto lo podemos conseguir siempre introduciendo dicha celda (o más, si se desea) mediante un comando `\multicolumn{1}{>{\columncolor...}}`.

Las siguientes declaraciones permiten controlar el color de los filetes de separación.

<code>\arrayrulecolor[<i>Modelo</i>]{<i>Color</i>}</code>	<code>\doublerulesepcolor[<i>Modelo</i>]{<i>Color</i>}</code>
---	---

Los argumentos siguen las mismas reglas que los dos primeros argumentos de `\columncolor` (*Color* es una especificación o un nombre de color, según esté presente *Modelo* o no).

EJEMPLO 3.18

```
\begin{center}\deactivatequoting
\newcommand*{\MiR}[2][gray]{%
\doublerulesepcolor[#1]{#2}}
\newcommand*{\MiLinea}{\hhline{|:|:|}}
>{\arrayrulecolor[gray]{.4}\MiR{.8}}=%
>{\arrayrulecolor[gray]{.8}\MiR{.4}}=%
>{\arrayrulecolor[gray]{.4}\MiR{.8}}=%
>{\arrayrulecolor{Black}\MiR{.4}}:|}%
\MiR{.4}
\begin{tabular}{||ccc||}\hhline{|t::=:t|}%
1 & 2 & 3 \\ \MiLinea a & b & c \\
\hhline{|b::=:b|}\end{tabular}\par\bigskip
\arrayrulewidth 3pt \arrayrulecolor{gray}
\doublerulesep 3pt \doublerulesepcolor{Yellow}
\begin{tabular}{||l||r||}\hline
alfa & beta \\
gamma & delta \\
\hline\end{tabular}\activatequoting\end{center}
```

1	2	3
a	b	c

alfa	beta
gamma	delta

Filetes dobles que cambian de color. Observe que hemos necesitado ejecutar `\deactivatequoting`, debido a una incompatibilidad, que sepamos desconocida hasta ahora, con la opción `spanish` de `babel`; sin ella la combinación de `\arrayrulecolor` y `\hhline` no funciona

La declaración `\arrayrulecolor` establece el color de los filetes de un `tabular` o `array`. Puede ser ubicado antes de iniciar uno de esos entornos y, en tal caso, afectará a todos los filetes, tanto horizontales como verticales. Si se introduce dentro del entorno ha de hacerse al comienzo de una fila, y afectará sólo a los filetes horizontales que sigan a la declaración (los verticales son asignados al inicio del entorno, por tanto no se ven afectados). Es posible, aunque no parece de gran interés, obtener filetes verticales en alguna fila de distinto color al de los mismos en el resto de la tabla, pero para ello debemos recurrir a `\color` y `\vrule` (véase el apartado 3.5.1).

Con `\doublerulesepcolor` se especifica el color que se utilizará en el espacio que hay entre dos filetes (horizontales o verticales) que forman un filete doble. Se le aplican los mismos comentarios que a `\arrayrulecolor`.

Los dos comandos anteriores son perfectamente compatibles con el comando `\hhline`, del paquete `hhline`. Más aún, podemos incluir los caracteres que determinan los entrelazamientos de filetes, siempre que tengamos cargado `hhline`, incluyendo declaraciones de color en el argumento de `\hhline`, como en la primera de las tablas del ejemplo siguiente. El comportamiento del paquete respecto al comando `\cline` sin embargo no es bueno. Los filetes que produce `\cline` serán prácticamente «tapados» por cualquier panel de color que se inserte en la fila inferior a ellos,

ya sea proveniente de un `\columncolor` o de un `\rowcolor`. Para obtener filetes de este tipo es enormemente ventajoso recurrir a `\hhline{-}` y similares, que no tienen ese problema.

Cuando un elemento de una tabla tiene un tamaño grande es muy probable que toque la línea horizontal precedente. En estos casos, podemos asignar un valor positivo a la longitud

`\minrowclearance`

consiguiendo un mejor efecto. Es similar a la longitud `\extrarowheight` del paquete `array`.

En el apartado PARA SABER MÁS al final del capítulo encontrará más información sobre otro tipo de tablas.

3.4. Sobre los objetos flotantes

QUIZÁS el lector ya ha tenido experiencias con los objetos flotantes de L^AT_EX, figuras y cuadros, y quizás estas experiencias han sido desagradables o, al menos, repletas de sorpresas. ¿Ha experimentado cómo un cambio, a veces mínimo, en el texto ha provocado un cambio radical en la posición de estos objetos? ¡Es algo así como el «efecto mariposa»! Esta sección se abre con la descripción de los diecisiete parámetros que L^AT_EX utiliza para ubicar los objetos flotantes. Su conocimiento puede ayudarle a influir sobre esta ubicación.

El resto de esta larga sección consiste en la exposición de diversos paquetes orientados al manejo de figuras y cuadros. Se refieren, entre otros aspectos, a más herramientas para la ubicación, control y mejora de las leyendas, etc. Uno de los problemas más interesantes es la ubicación de figuras rodeadas de texto o marginadas, es decir, figuras que, en lugar de utilizar todo el ancho del texto, se colocan en una caja dentro de un párrafo, rodeada ésta de texto por sus aristas superior o inferior y una de las laterales.

3.4.1. Control de la ubicación

Sabemos que podemos influir en la ubicación de los objetos flotantes incluyendo el argumento optativo *Posición* de los entornos `figure` y `table`. Recordemos que este argumento consiste en uno o más caracteres de entre `h`, `t`, `b`, `p` y `!`. Recordemos también que el orden en el que se escriban estos caracteres es indiferente y que, si el argumento *Posición* no se incluye, L^AT_EX se comporta como si se hubiera incluido con el valor `tbp` (véase el apartado I.9.4).

L^AT_EX recurre al siguiente algoritmo recursivo hasta que consigue colocar el objeto flotante:

1. Un entorno flotante no se imprime antes de otro de la misma familia que le precede en el fichero fuente (esto vale para los entornos `figure` y `table`, pero puede no ser cierto para entornos análogos implementados en paquetes específicos).
2. Un entorno `figure` o `table` no puede ubicarse si se excede el tamaño restante de la página, teniendo en cuenta el texto ya ubicado en dicha página.
3. Si aparece `h` en *Posición* intenta colocar la figura exactamente en esa posición. Si no es posible y no se ha especificado otro valor, asigna el valor `ht`, indicando este cambio de criterio con un mensaje en pantalla, que se guarda en el fichero `log`. A partir de aquí el algoritmo se detendrá si en uno de los pasos siguientes se consigue ubicar al objeto flotante.

4. Si se especifica `t` trata de colocarlo al comienzo de la página actual.
5. Si se especifica `b` trata de colocarlo al final de la página actual.
6. Si se especifica `p` trata de colocarlo, cuando termine la página actual, en una página que sólo contiene objetos flotantes.
7. Los pasos 4 y 5 se repiten, si todavía no se ha conseguido ubicar el objeto, al comienzo de cada página subsiguiente; el paso 6 se repite al final de dicha página, y todo ello hasta el final del documento.

Estas reglas se anulan cuando el compilador encuentra uno de los comandos `\clearpage`, `\cleardoublepage` o `\end{document}`, en cuyo caso todos los flotantes que no se han ubicado con anterioridad se imprimen, en ese momento, en las páginas que sean necesarias (de tipo `p`).

Los contadores y comandos presentados a continuación establecen parámetros que L^AT_EX tiene en cuenta para la ubicación de los objetos flotantes. Entre paréntesis se indica el valor por defecto de cada uno de ellos, salvo para aquellos cuyo valor depende de la opción de tamaño del texto normal (10pt, etc.).

Contadores sobre el número de objetos flotantes en una página

`topnumber` Indica el número máximo de objetos flotantes que pueden imprimirse en la parte superior de la página (2).

`dbltopnumber` Tiene el mismo significado que `topnumber` pero para flotantes compuestos a todo el ancho de la mancha, es decir, con `figure*` o `table*`, en texto a dos columnas (2).

`bottomnumber` Indica el número máximo de objetos flotantes que pueden imprimirse en la parte inferior de la página (1).

`totalnumber` Es el número máximo de objetos flotantes que pueden imprimirse en una página de texto normal, sin tener en cuenta su localización (3).

Comandos sobre el reparto de objetos flotantes y texto

`\topfraction` Es un número decimal menor que 1 que indica la fracción máxima de la página que puede ser ocupada por objetos flotantes colocados en la parte superior de la misma (0,7, es decir, 70 %).

`\dbltopfraction` Como `\topfraction` para flotantes a dos columnas (0,7).

`\bottomfraction` Es un número decimal menor que 1 que indica la fracción máxima de la página que puede ser ocupada por flotantes ubicados en la parte inferior de la misma (0,3).

`\textfraction` Es un número decimal menor que 1 que indica la fracción mínima de una página normal que tiene que completarse con texto (0,2).

`\floatpagefraction` Es un número decimal menor que 1 que indica la fracción mínima de una página de objetos flotantes que tiene que completarse con tales objetos (0,5).

`\dblfloatpagefraction` Tiene el mismo significado que `\floatpagefraction` para flotantes a dos columnas (0,5).

Longitudes de separación

Todas las longitudes que siguen son elásticas.

\textfloatsep La separación entre un objeto flotante, colocado en la parte superior o inferior de la página, y el texto que le sigue o que le precede.

\dbltextfloatsep El mismo significado que \textfloatsep para flotantes a dos columnas.

\floatsep Es la separación vertical entre dos objetos flotantes consecutivos, ambos colocados en posición t o b.

\dblfloatsep Tiene el mismo significado que \floatsep para flotantes a dos columnas.

\intextsep Es la separación vertical entre un objeto flotante y el texto que le precede y que le sigue cuando se ha ubicado en la posición h.

\abovecaptionskip Longitud del espacio vertical extra insertado antes de colocar la leyenda del elemento flotante (10 pt).

\belowcaptionskip Longitud del espacio vertical extra insertado después de colocar la leyenda del elemento flotante (0 pt).

Los parámetros anteriores pueden modificarse a fin de influir en el resultado del algoritmo que L^AT_EX emplea para ubicar los objetos flotantes, pero conviene no olvidar que no son parámetros independientes. Las modificaciones tendrán efecto a partir de la página siguiente a aquella en la que se realizan.

En documentos con muchos objetos flotantes puede ocurrir que haya varias páginas que únicamente contienen objetos flotantes y además que en tales páginas exista mucho espacio en blanco; ello es debido al valor que por defecto tiene \floatpagefraction y al algoritmo que L^AT_EX utiliza. La modificación de este parámetro puede ser de utilidad para la ubicación de un objeto flotante, pero puede provocar efectos secundarios indeseables en otros. Por ejemplo, aumentar dicho parámetro evitará tener páginas de flotantes con mucho blanco, pero también hará más difícil ubicarlos en este tipo de páginas, produciendo acumulación de flotantes pendientes de colocación. Esta acumulación es peligrosa; es fácil encontrar que todos los flotantes, a partir de un cierto momento, terminan en páginas de flotantes al final del documento (o del capítulo). En caso de encontrarse con esta situación trate de incluir o modificar el argumento *Posición* del primero de ellos, hasta conseguir que sea ubicado en un lugar razonablemente próximo al que ocupa en el documento fuente.

En general, es aconsejable dejar que L^AT_EX sitúe los objetos flotantes por sí mismo e incluir el argumento opcional en aquellos casos en los que el resultado producido no nos agrade. En esas ocasiones, la simple inclusión del parámetro optativo h, b, t puede conseguir el efecto deseado. Si todavía no lo consigue utilice el carácter !, mediante el cual conseguimos que se ignoren todos los parámetros de los dos primeros grupos. Este modulador de la posición no afecta a los flotantes ubicados en páginas que sólo contienen objetos flotantes. Un objeto que posee el modulador ! en su argumento *Posición* será siempre colocado en alguna de las posiciones indicadas por este argumento, salvo que con ello se viole una de las reglas 1 o 2 del algoritmo mostrado al inicio de este apartado. Si todas las estrategias fallan, siempre podemos recurrir a un truco elemental, aunque no muy elegante: cambiar de posición en el fichero fuente el objeto flotante; a veces es lo más rápido

y efectivo, nuestra experiencia lo confirma, pero tiene el inconveniente de ser totalmente manual y, por tanto, necesita revisiones en caso de que se modifique el texto.

Tenga en cuenta que las limitaciones del espacio ocupado por los objetos flotantes consideran la longitud `\textfloatsep` como parte del flotante.

Con ayuda del paquete `float` (véase el apartado PARA SABER MÁS al final de esta sección) y haciendo uso del valor `H` puede influirse de forma directa sobre la ubicación de objetos flotantes.

Observando los valores por defecto de los parámetros anteriores, parece que L^AT_EX prefiriera ubicar los objetos flotantes en la parte superior de la página, y así es. En algunos casos eso puede hacer que un objeto flotante aparezca impreso antes de que se hable de él. Para evitar esta situación es posible utilizar el comando

<code>\suppressfloats[Posición]</code>
--

donde el argumento optativo *Posición* puede ser `t` o `b`. La utilización de este comando, en cualquier parte del documento, hace que en esa página, en el área especificada por *Posición*, o en cualquier área si no se incluye el parámetro optativo, no se impriman los objetos flotantes.

Elementos de estilo

Los siguientes comandos permiten modificar ligeramente la forma en que se componen los objetos flotantes:

<code>\topfigrule</code>	<code>\botfigrule</code>	<code>\dblfigrule</code>
--------------------------	--------------------------	--------------------------

El primero se ejecuta después de colocar un objeto flotante en la parte superior de una página. El segundo se ejecuta antes de ubicar un objeto flotante en la parte inferior de una página. El tercero es análogo a `\topfigrule` pero referido a objetos flotantes a dos columnas. Por defecto ninguno de ellos «hace nada» (su definición es vacía), pero pueden ser redefinidos como se deseé. Su objetivo, como indican los nombres, podría ser la inclusión de algún filete. L^AT_EX espera que el resultado de su ejecución ocupe un espacio vertical nulo. Así podríamos, por ejemplo, redefinir:

```
\renewcommand*{\topfigrule}{\vspace*{-1pt}%
\rule{\columnwidth}{0.4pt}\vspace{.6pt}}
```

3.4.2. Incluir objetos al comienzo de una página: el paquete `afterpag`

Como acabamos de comentar, a veces L^AT_EX encuentra dificultades para colocar los objetos flotantes de acuerdo con la intención del autor del manuscrito. Ya sabemos que existe la posibilidad de provocar que todos los objetos flotantes que aún no se han ubicado lo hagan inmediatamente, mediante el comando `\clearpage`, cuyo efecto es doble: produce un salto de página con una finalización prematura de la página actual y ubica en la página siguiente los objetos flotantes que todavía estaban sin colocar.

El paquete `afterpag`, desarrollado por David Carlisle [8], que proporciona medios alternativos de ubicación, implementa el comando de un argumento:

<code>\afterpage{Argumento}</code>

cuyo efecto es que el *Argumento* se procesará después de completar la página actual. Aunque el comando puede utilizarse con finalidades muy diferentes, pondremos dos ejemplos sobre objetos flotantes, en los que resulta útil.

El primero se refiere a una utilización más flexible del comando `\clearpage` al que antes nos hemos referido. Mediante

```
\afterpage{\clearpage}
```

se consigue que el comando `\clearpage` entre en acción en el momento de completarse la página en la que se introduce dicha orden. Esto significa que no se generará un nuevo salto de página (pues `\clearpage` es ignorado al comienzo de una página), pero sí se ubicarán los flotantes pendientes.

El segundo ejemplo tiene por objeto evitar que una tabla larga, con más de una página de longitud, construida con un entorno `longtable` (véase la sección 3.3.3), se inicie a mitad de una página. Para ello se escribe la tabla en un fichero externo, digamos `tabla.tex`, y en lugar de que el fichero principal la inserte en un sitio preciso, se la deja «flotar» hasta que se inicie una nueva página mediante el uso de una de las dos siguientes formulaciones:

```
\afterpage{\clearpage\input{tabla.tex}}
```

```
\afterpage{\clearpage\input{tabla.tex}\clearpage}
```

En la primera formulación, la tabla comienza en una nueva página, pero al acabar la tabla continúa el texto. En la segunda, el texto no continúa hasta la página siguiente.

3.4.3. Muchos formatos de leyendas con el paquete `caption2`

`LATEX` compone la leyenda de una figura o cuadro sólo en dos formas, según la longitud de dicha leyenda. O bien está centrada, en caso de que su longitud no llegue a superar el ancho de línea, o bien se compone en un párrafo sin sangría, en caso contrario. En ambas situaciones utiliza un tipo de letra normal (la misma que en el texto usual, dada por `\normalfont`). El paquete `caption2`, desarrollado por Haral Axel Sommerfeldt [59], implementa opciones y herramientas para personalizar las leyendas que aparecen en los entornos `figure` y `table` (entre otros). Como es lógico, no todas las opciones son compatibles entre sí. En el cuadro 3.6 se muestran agrupadas según el aspecto a que se refieren, siendo incompatibles entre sí las de cada grupo, así como las separadas por una barra vertical.

Como se observa en las últimas opciones, el paquete puede soportar, además de los entornos `figure` y `table`, los entornos creados con el paquete `float` así como los implementados en los paquetes `longtable` y `subfigure`. El orden utilizado para cargar estos paquetes puede influir, y es recomendable cargar `caption2` en primer lugar.

La simple utilización de estas opciones ya permite un margen notable de flexibilidad en el formato de la leyenda. Con ayuda de las herramientas que se describen a continuación es posible incrementar dicha flexibilidad. Además de `\abovecaptionskip` y `\belowcaptionskip`, que ya hemos comentado en la página 303, los siguientes comandos, introducidos en el paquete, permiten un mejor control de la caja en la que se componen las leyendas.

<code>\setcaptionwidth{Longitud}</code>	<code>\setcaptionmargin{Longitud}</code>
---	--

Opción	Descripción
normal	Corresponde al formato usual de LATEX. Es la que utiliza por defecto el paquete si no se declara ninguna de las opciones que aparecen en este grupo.
centerlast	El formato es el mismo que con la opción normal, sólo que la última línea aparece centrada, es decir, en «triángulo español».
hang	La leyenda aparece como si se tratara de un ítem cuya etiqueta fuera el antetítulo de la misma, es decir, \figurename\thefigure (lo análogo para el caso de cuadros). La etiqueta (es decir, la primera línea) no se sangra.
indent	Similar a la anterior, con la diferencia de que en aquélla el texto de la leyenda se situaba estrictamente a la derecha de la leyenda, mientras que en ésta las líneas posteriores a la primera (si las hubiere) se sangran una longitud igual al valor de \captionindent.
center	Todas las líneas de la leyenda aparecen centradas.
flushleft flushright	Como la anterior, pero en bandera por la derecha o por la izquierda.
hang+center	Son opciones que resultan en la conjunción de hang con cada una de las tres anteriores. Incluir dos opciones, como hang y center, sólo dará como resultado que se tome la que figure en la última posición.
hang+centerlast	
hang+flushleft	
scriptsize footnotesize small normalsize large Large	Indica el tamaño de los tipos con los que se imprimirá la leyenda.
rm sf tt	Selecciona la familia de tipos y afecta únicamente a la etiqueta de la leyenda.
up it sl sc	Como la anterior, pero selecciona el perfil.
md bf	Como la anterior, pero selecciona el grosor o serie.
oneline nooneline	Cuando la leyenda contiene una sola línea, entonces, por defecto, aparece centrada (lo que corresponde a la opción oneline). La opción nooneline evita este comportamiento y compone todas las leyendas de la misma forma.
longtable	Adapta las leyendas del paquete longtable a las usuales, con el formato especificado por las opciones de caption2.
float	Como la anterior respecto al paquete float (véase el apartado PARA SABER MÁS al final de esta sección).
subfigure	Como las anteriores respecto al paquete subfigure (v. PARA SABER MÁS).
all none	Activa o anula la adaptación a los tres paquetes anteriores.
debug	Incluye información de las acciones realizadas por el paquete en el fichero log.

Cuadro 3.6: Opciones del paquete caption2

El primero de ellos establece el valor de la longitud \captionwidth como igual a *Longitud*, esta longitud es la anchura absoluta en la que se compondrán las leyendas. El segundo establece otra longitud, \captionmargin, que es una longitud extra que se añade a los márgenes izquierdo y derecho de la leyenda. El efecto de estos dos comandos no se superpone, es decir; si se ha establecido una anchura absoluta, mediante \setcaptionwidth y, después, se establecen unos márgenes, me-

diente `\setcaptionmargin`, será la anchura calculada por este último comando la que prevalecerá, y dicha anchura es la que resulta de restar dos veces la establecida por `\setcaptionmargin` a `\ linewidth` y no a la establecida por `\setcaptionwidth`. A estas longitudes hay que añadir otra

`\captionindent`

ya comentada en el cuadro 3.6 a propósito de la opción `indent`.

El paquete `caption2` proporciona además la posibilidad de definir, y, por supuesto, utilizar posteriormente, nuevos estilos de leyenda. Los comandos fundamentales son:

`\newcaptionstyle{NombreEstilo}{Definición}`
`\renewcaptionstyle{NombreEstilo}{Definición}`

Mediante el primero de ellos se define el estilo llamado *NombreEstilo*; su definición debe ser introducida en el argumento *Definición*. El segundo de los comandos es semejante, pero su objetivo es cambiar la definición del estilo *NombreEstilo* ya existente. A este respecto es interesante hacer observar que cada una de las opciones del paquete incluidas en los dos primeros grupos, desde `normal` hasta `hang+flushleft`, define un estilo de nombre exactamente igual al de la opción; esto significa que podríamos redefinir cualquiera de esos estilos.

Una vez definido un nuevo estilo, llamado *NombreEstilo*, para poder utilizarlo en todas las leyendas debemos incluir la declaración

`\captionstyle{NombreEstilo}`

Hay que tener en cuenta que las opciones del paquete no afectarán a estos estilos definidos por el usuario. De hecho, estas opciones lo que hacen es definir sus propios estilos. Si quisieramos un estilo similar a `hang` debemos construirlo «a mano», mediante una buena *Definición*.

Como se observa, la dificultad que subyace en este proceso se centra, naturalmente, en el argumento *Definición* de `\newcaptionstyle`. Los siguientes comandos son herramientas preparadas para que el usuario pueda llevar a cabo esta tarea con más comodidad.

`\captionlabel` `\captiontext`

El primero de ellos imprime el antetítulo (es decir, `\figurename`~`\thefigure`, y lo análogo para los cuadros). El segundo hará lo mismo con el texto de la leyenda propiamente dicho (es decir, el argumento de `\caption`).

A éstos hay que añadir otros comandos y longitudes:

`\captionlabeldelim` `\captionlinewidth`
`\captionlabelsep` `\usecaptionmargin`

`\captionlabeldelim` contiene el delimitador que se utiliza después del antetítulo de la leyenda y que, usualmente, tiene por valor el carácter dos puntos (:). Puede ser redefinido, con `\renewcommand`, en cualquier momento.

`\captionlabelsep` es un comando que inserta un espacio entre `\captionlabeldelim` y el comienzo del texto de la leyenda. No se trata de una longitud, sino de un comando; por tanto, debe ser redefinido con `\renewcommand` y, usualmente, consistirá en un comando que inserte un cierto espacio.

\captionlinewidth es la anchura final que tendrá la primera línea de la leyenda (téngase en cuenta que, según las opciones, la anchura de las otras líneas puede ser distinta). Puede ser útil para colocar objetos dentro de la leyenda.

\usecaptionmargin utilizado dentro del argumento *Definición* consigue que se tengan en cuenta la longitud establecida por \setcaptionwidth, o por \setcaptionmargin (según cuál esté activa, en el sentido comentado antes). Así conseguiremos que nuestros nuevos estilos puedan mantener abierta esta posibilidad de modificación de la anchura del texto y, además, mantengan la misma que las leyendas de otros estilos que, eventualmente, puedan aparecer en el mismo documento.

El siguiente es un ejemplo sencillo del modo en que puede crearse un nuevo estilo de leyenda y la forma de usarlo con ayuda del comando \newcaptionstyle.

EJEMPLO 3.19

```
\setcaptionwidth{5cm}
\renewcommand*\figurename{Fig.}
\renewcommand*\captionlabeldelim{.}
\renewcommand*\captionlabelsep{\hspace{.5em}}
\newcaptionstyle{MiCaption}%
  {\usecaptionmargin\centerline{%
    \mbox{\scshape\figurename\ \thefigure}%
    \captionlabeldelim\captionlabelsep%
    \parbox[t]{\captionlinewidth}{%
      \captiontext}}}
\captionstyle{MiCaption}
\begin{figure}
\includegraphics[width=3cm]{paloma.pcx}
\caption{Una de las múltiples representaciones de la paloma de la paz}
\end{figure}
```

Nuevo estilo de leyenda con el paquete caption2, similar a hang



FIG. 3.2. Una de las múltiples representaciones de la paloma de la paz

En la página 85, nota 5, advertíamos del mal comportamiento en las leyendas vacías de figuras o cuadros: recuerde que si incluimos \caption{} obtendremos una leyenda que acaba con el signo de dos puntos, al que no sigue texto alguno. En la citada página prometíamos una solución; hela aquí. Cargaremos el paquete caption2 con opción normal (si deseamos mantener las leyendas en la forma estándar de L^AT_EX). Llegado el momento en que sea necesario incluir un comando \caption con argumento vacío, procedemos a redefinir el comando \captionlabeldelim antes de \caption, pero dentro del entorno flotante al que corresponde dicho \caption. Lo natural será redefinir \captionlabeldelim como un punto o, casi mejor, como vacío. Observe que al redefinirlo dentro del entorno flotante, cuyo contenido forma un grupo, no tenemos que volver a la definición primitiva del delimitador pues el cambio realizado es sólo local.

Finalmente podemos recurrir a otros comandos que, aunque son principalmente de uso interno del paquete, su empleo no plantea problemas:

\captionlabelfalse	\captionfont
\captionlabeltrue	\captionlabelfont

Incluyendo el primero en algún lugar del documento fuente, las leyendas que sigan no contendrán el antetítulo ni el número de orden del entorno. Este comportamiento se puede anular con el contrario `\captionlabeltrue`, que es la declaración por defecto.

Los comandos `\captionfont` y `\captionlabelfont` contienen los atributos de los tipos con los que se escribirán las leyendas. Se definen internamente en función de las opciones del paquete referentes al tamaño, perfil y grosor. El primero toma el valor de las opciones sobre el tamaño de los tipos y afecta a toda la leyenda. El segundo depende de las opciones del perfil y el grosor y sólo afecta a la etiqueta de la leyenda. Pueden ser redefinidos en cualquier momento. Si queremos que `\captionlabelfont` afecte a toda la leyenda, y no sólo a su etiqueta, podemos redefinir `\captionfont` en la forma siguiente:

```
\let\OrigCaptionFont=\captionfont
\renewcommand*\captionfont{\captionlabelfont\OrigCaptionFont}
```

Es preciso señalar que los comandos anteriores no afectarán a las leyendas compuestas con los estilos definidos o redefinidos por el usuario, a través de `\(re)newcaptionstyle`. Podemos, sin embargo, incluir las opciones sobre fuentes del paquete que deseemos, lo que define internamente los comandos anteriores y, a la hora de definir un estilo, utilizar éstos en el argumento *Definición* de `\newcaptionstyle`. Otro inconveniente ocurre con las opciones `oneline|nooneline`. Sería deseable que estas opciones afectaran a todos los estilos, incluyendo los nuevos, pero no es así. Esto se debe a la forma en que están implementadas estas opciones (y los comandos anteriores) en el paquete y, lo que no pasa de ser una curiosidad graciosa, tiene que ver con comandos que incluyen en su nombre dos o tres arrobas (@). Podemos hacer que todas estas cosas afecten también a los nuevos estilos con el siguiente código:

```
\newcaptionstyle{Estilo1}{Definición1}
\makeatletter
\newcaptionstyle{Estilo2}{\onelinecaption{Definición2}\caption@@Estilo1}
\makeatother
\captionstyle{Estilo2}
```

En la *Definición1* incluiríamos la forma en que se debe componer la leyenda si su ancho total es mayor que `\captionlinewidth`, mientras que la *Definición2* contendría la definición en caso contrario; de esta forma la leyenda corta será centrada, el comando

`\onelinecaption`

se encarga de la comprobación sobre la anchura y de centrar en caso necesario. En estas definiciones podemos utilizar todos los comandos relacionados anteriormente.

3.4.4. Pequeñas figuras rodeadas de texto

A menudo los objetos flotantes, sobre todo algunas figuras, tienen una anchura mucho menor que la de la caja del texto. Su inclusión como objetos flotantes los coloca entonces a todo el ancho de esta caja, exceptuando el caso de `figure` y `table` con texto a dos columnas, lo que produce una pérdida considerable de espacio e incluso, a veces, un efecto estético cuestionable. Para estos

objetos pequeños podría ser recomendable su inclusión en un párrafo, quedando circundados por el texto ordinario, ya sea por dos o tres lados del objeto; es lo que se denomina «figuras marginadas», utilizando técnicamente el término *arracada* para denominar al nicho blanco que se reserva en el texto para la inclusión del objeto. La realización de esta construcción con L^AT_EX no es sencilla o, al menos, no es sencillo conseguirla mediante los automatismos adecuados, es decir, sin que el usuario deba contar, por ejemplo, el número de líneas cortas que sumarán una altura similar a la del objeto.

En el fondo el problema es el mismo que el de conseguir párrafos con formas extrañas, como el que se ilustra en la figura 8.1, página 474, o como el que aparece en el ejercicio I.3.4 de la lección 3 (véase también el comando \hangafter descrito en la página 473). En esta sección exponemos las herramientas que proporciona el paquete picins para la inclusión de estas pequeñas figuras. Existen también otros paquetes que se ocupan de este tema y, en general, todos presentan dificultades y resultados deficientes en algunas circunstancias, lo cual concuerda con nuestra anterior afirmación sobre lo complicado que resulta resolver este problema con un procedimiento automatizado.

Desarrollado por Joachim Bleser [4], el paquete picins produce resultados bastante buenos, aunque tiene ciertas limitaciones. El comando fundamental proporcionado por el paquete es:

\parpic(<i>Ancho</i> , <i>Alto</i>) (<i>DesplH</i> , <i>DesplV</i>) [<i>Opciones</i>] [<i>Posición</i>] { <i>Objeto</i> }

Como se observa, la sintaxis no es la usual de los comandos de L^AT_EX, fundamentalmente por los paréntesis (aunque los paréntesis, como delimitadores de argumentos también aparecen en algunos comandos de L^AT_EX, concretamente en el entorno picture, véase el apartado PARA SABER MÁS del capítulo 4).

El argumento *Objeto* será lo que se coloque como «objeto marginado» en el párrafo, rodeado de texto por los lados inferior e izquierdo o derecho (según las *Opciones*). Éste es el único argumento obligatorio de \parpic y puede consistir en cualquier tipo de objeto: una caja, un gráfico incluido con \includegraphics, etc.

El comando \parpic inicia un párrafo, lo que significa que sólo es posible incluir el *Objeto* al inicio del párrafo que lo circunde o, dicho de otra forma, el *Objeto* no podrá tener líneas del párrafo en el que se inserta por encima de él. Existen soluciones alternativas que, sin incumplir esta condición, producen un resultado equivalente a la existencia de líneas del párrafo por encima del *Objeto* (véase el ejemplo 8.10).

A continuación se detalla el significado de otros argumentos de \parpic.

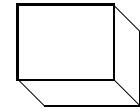
Opciones Consiste en uno o dos caracteres, cada uno de ellos elegido en uno de los grupos siguientes:

l, r: indica el lado del párrafo en que se insertará el *Objeto*, l a izquierda, r a derecha. Las declaraciones

\picchangemode	\nopicchangemode
----------------	------------------

alteran este comportamiento. La primera invierte las opciones l y r en las páginas a izquierda, lo que significa que l y r pasan a significar en la parte interior y exterior del texto, respectivamente. La segunda declaración anula a la primera.

f, d, o, s, x: su efecto es el de incluir un marco alrededor del *Objeto*; cada carácter especifica un tipo de marco: **f** un marco usual (del tipo `\fbox`), **d** un marco realizado con líneas discontinuas, **o** un óvalo, **s** un marco sombreado y **x** un marco de tipo tridimensional (un cubo), como el marginado en este mismo párrafo.



El orden en el que aparecen los dos caracteres, si los hay, es irrelevante. El valor por defecto, es decir, el que se toma si no se incluye el argumento *Opciones*, es 1.

(Ancho, Alto) Asigna la anchura y la altura de la caja reservada para la inclusión del *Objeto*. Tanto *Ancho* como *Alto* son, por tanto, longitudes. Si alguna de estas longitudes no se incluye o si es nula, la dimensión correspondiente de la caja será la natural del *Objeto*.

(DesplH, DesplV) Indica un desplazamiento horizontal de *DesplH* y vertical de *DesplV* respecto a la posición natural del *Objeto*. Si se utiliza este argumento también se debe incluir **(Ancho, Alto)**, con *Ancho* y *Alto* no vacíos. Éstos pueden ser nulos, pero en este caso el *Objeto* se desplazará respecto a una caja que tiene exactamente las dimensiones del *Objeto*, lo que supondrá que éste se superponga al texto circundante.

Posición Consiste en uno o dos caracteres, uno de ellos igual a l (izquierda) o r (derecha), el otro igual a t (superior) o b (inferior). Indica la posición del *Objeto* dentro de la caja especificada por **(Ancho, Alto)** y sólo tiene sentido si esta caja es mayor que la caja que ocupa *Objeto*. Si no se indica este argumento la posición es centrada (lo que corresponde al valor c para este argumento). Si se incluye el argumento **(DesplH, DesplV)**, éste tiene prioridad sobre *Posición*, es decir, este último no se considera, por lo que es innecesario.

Los siguientes comandos permiten personalizar algunos de los elementos gráficos descritos anteriormente:

<code>\dashlength{Long}</code>	<code>\shadowthickness{Long}</code>	<code>\boxlength{Long}</code>
--------------------------------	-------------------------------------	-------------------------------

El primero de ellos asigna el valor del argumento *Long*, que debe ser una longitud, a la longitud de los segmentos que forman las cajas de aristas discontinuas (*Long* se envía como argumento a `\dashbox`, véase el apartado PARA SABER MÁS del capítulo 4). `\shadowthickness` establece el grosor de la sombra de las cajas especificadas por el valor **s** en *Opciones* y `\boxlength` asigna *Long* como la profundidad de las cajas tridimensionales.

El siguiente comando define la separación horizontal entre la caja marginada y el texto:

<code>\pichskip{Longitud}</code>

Al comienzo de esta sección advertíamos de ciertos problemas que se presentan en estas «cajas marginadas». ¿Cuáles son estos problemas? ¿Podemos resolverlos? Fundamentalmente los problemas son comunes a todos los paquetes que persiguen este objetivo y se presentan, en general, en dos situaciones: cuando el párrafo en el que se inserta el *Objeto* se rompe al final de una página y cuando cerca del *Objeto* se inserta algún entorno que cambia la anchura de la línea de texto. La respuesta de `picins` en estas situaciones es relativamente mejor que la de otros paquetes.

- La primera de las situaciones es inevitable. El *Objeto* insertado no tiene el carácter de flotante; por tanto, L^AT_EX no tiene más remedio que enviarlo a la página siguiente, si no cabe en la actual, lo que, probablemente, dará una página corta. Pero, a veces, el párrafo se rompe a

mitad y el *Objeto* sobresale por debajo de las líneas del párrafo que han quedado junto a él. En estos casos no queda más remedio que recurrir a algunas herramientas de control de la rotura de páginas (`\pagebreak`, `\enlargethispage`, etc.).

- La segunda situación es más problemática, porque los resultados pueden ser muy malos. En general el *Objeto* se superpondrá con el texto si, cuando las líneas cortas del párrafo en el que se ha incluido no han llegado a cubrir toda la altura del *Objeto*, se incluye un entorno que altere la anchura de la línea de texto; por ejemplo, cualquier entorno de tipo lista (`enumerate`, `quote`, etc.). En este caso una solución puede ser aumentar artificialmente la anchura de la caja destinada a incluir el *Objeto*, mediante el argumento (*Ancho*, *Alto*) de `\parpic`, o utilizar `\pichskip` para aumentar la distancia horizontal entre caja y texto, de forma que las líneas serán más estrechas y quizás agoten así la altura del *Objeto*. Naturalmente otra posibilidad, más compleja, es modificar, si es posible, los parámetros (márgenes) de la lista, en la forma indicada en el apartado 3.2.2 o, con un poco de trabajo de medición, insertar el entorno `itemize`, `quote`, etc., en una caja `\parbox` o `minipage` de la anchura apropiada, la misma que las líneas estrechas del párrafo (es decir, `\columnwidth` menos la anchura de la caja que contiene al *Objeto*, menos la asignada por `\pichskip`).
- Una situación en la que `picins` sí responde netamente mejor que otros paquetes es la que se presenta cuando queremos circundar un *Objeto* mediante los ítems de una lista. La respuesta es buena siempre y cuando la altura total de la lista supere la de la caja reservada para el *Objeto*. Para conseguir que esto funcione debemos incluir el comando `\parpic` inmediatamente antes o inmediatamente después de un comando `\item` de la lista.
- Normalmente no se presenta problema alguno si el párrafo en que se inserta el *Objeto* contiene una fórmula matemática resaltada (en estilo `display`), aunque en algunas ocasiones es preciso recurrir a `\pickskip`, como se explica a continuación.

En algunas ocasiones el paquete `picins` se «equivoca» y no hace estrechas tantas líneas del párrafo como hace falta para enmarcar el *Objeto* con el texto; normalmente se queda corto, y la parte inferior del *Objeto* se superpone a una o más líneas. Producida esta situación podemos ayudarle contando las líneas que nos harían falta, más o menos, e insertando el comando

`\pickskip{Número}`

A partir de este comando se harán exactamente tantas líneas cortas como indica *Número*. Para que surta efecto debe ser colocado después de `\parpic` (puesto que éste hace sus cálculos sobre el número de líneas cortas, la asignación de `\pickskip`, realizada antes de `\parpic`, es anulada por éste). Tenga en cuenta los espacios verticales que puedan introducir algunos comandos o entornos (por ejemplo, una fórmula centrada conviene contarla como tres líneas). El comando inicia un párrafo y no debe ir seguido inmediatamente por otro inicio de párrafo.

Leyendas El paquete `picins` permite también poner una leyenda al *Objeto* insertado. El comando que se encarga de ello es

`\piccaption[TextoLeyendaÍndice]{TextoLeyenda}`

que, como se observa, tiene una sintaxis idéntica a la del comando `\caption` del entorno `figure` (véase el apartado I.9.4). De hecho, la leyenda incluida por `\piccaption` es exactamente la de un entorno `figure`, es decir, va precedida del antetítulo `\figurename` y del número de orden del entorno `figure`. Dicho de otro modo, cuando se incluye el comando `\piccaption` el *Objeto* insertado por `\parpic` se comporta exactamente como si de un entorno `figure` se tratase, excepto en su carácter flotante. Observe que este carácter no flotante puede conllevar una mala ordenación de las figuras, en caso de que, existiendo todavía algún entorno `figure` pendiente de ubicar, se incluya una con `\parpic`. Sin embargo, el comando se comporta de forma extraña respecto al índice de figuras: inserta automáticamente una entrada en dicho índice, pero no insertará *TextoLeyenda* a no ser que se incluya el argumento opcional *TextoLeyendaÍndice* (ignoramos si este particular es un error en el diseño del paquete o ha sido diseñado así voluntariamente).

Una observación fundamental sobre `\piccaption` es que debe ser incluido en el fichero fuente *antes* que el `\parpic` al que se refiere (lo natural es incluirlo inmediatamente antes). También es importante indicar que se puede incluir una etiqueta, mediante `\label`, para referencias a la figura, pero que, en ese caso, `\label` debe aparecer dentro del argumento *TextoLeyenda* de `\piccaption`.

Las siguientes declaraciones controlan la posición de la leyenda respecto al *Objeto*:

<code>\piccaptionoutside</code>	<code>\piccaptionside</code>
<code>\piccaptioninside</code>	<code>\piccaptiontopside</code>

`\piccaptionoutside` y `\piccaptioninside` colocan la leyenda en el exterior y en el interior, respectivamente, del marco que contiene al *Objeto*. Si no se introduce un marco, mediante el argumento *Opciones* de `\parpic`, estas declaraciones son irrelevantes. Ambas colocan la leyenda debajo del *Objeto*. El comando `\piccaptionside` coloca la leyenda al lado del *Objeto*, centrado verticalmente respecto a él, lo que suele destruir por completo el que el *Objeto* esté rodeado de texto; esta opción permite colocar los objetos con la leyenda a un lado ocupando el espacio que falta al *Objeto* para cubrir el ancho de la mancha, lo que proporciona la ventaja, frente a figuras a todo el ancho, de reducir el espacio vertical ocupado por la leyenda. `\piccaptiontopside` coloca la leyenda al lado del *Objeto* alineada verticalmente con la parte superior de la caja que lo contiene; el texto del párrafo circundante se desplazará hacia abajo las líneas que ocupe dicha leyenda.

Uno de los inconvenientes de picins respecto a las leyendas es que no posee una forma de establecer un *Objeto* insertado como un cuadro. Podemos alcanzar este objetivo cambiando el tipo de leyenda, que está almacenado en el comando `\@capttype`. Así el código

```
\makeatletter
\renewcommand{\@capttype}{table}
\makeatother
... \piccaption{...} \parpic{...}
```

colocará una leyenda de tipo `table` (cuadro) en el *Objeto* insertado por `\parpic`. Nos enfrentamos, sin embargo, a algunas dificultades en esta cuestión:

- necesitamos incluir `\makeatletter` y `\makeatother`, debido a la presencia de `@` en el nombre del comando redefinido (véase la página 217);

- si queremos tener varios `\parpic`, unos de tipo figura y otros de tipo cuadro, tenemos que redefinir, antes de cada cambio de tipo, el comando `\@partype`, ya que la redefinición anterior no parece que pueda hacerse en un grupo, con el fin de convertirla en local. Existe, sin embargo, una solución más elegante: definir un comando (`\MiPicCaption`, por ejemplo) que opta por un tipo u otro de leyenda, según el valor de su primer argumento; la definición podría ser la siguiente:

```
\newcommand*\{\MiPicCaption\}[1][figure]{\makeatletter
    \renewcommand*\{\@captive\}{#1}\makeatother\piccaption}
```

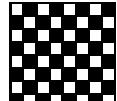
El paquete `picins` proporciona algunas otras herramientas, de menor envergadura, que no detallaremos aquí (consulte [4]). Efectos similares al conseguido con `picins` pueden obtenerse con el paquete `wrapfig` y con el fichero de macros para `TEX` llamado `Insbox.tex`. Del primero de ellos, que genera más problemas que `picins`, puede encontrar una descripción en [11]. El segundo funciona bastante bien, pero el usuario necesita trabajar más, puesto que las herramientas que contiene son más elementales y no utiliza los recursos de `LATEX`.

En el apartado PARA SABER MÁS, al final del capítulo, se menciona otro paquete útil para este propósito, el paquete `picinpar`. Si necesita numerar subfiguras en un entorno `figure` o si con dos entornos flotantes no tiene suficiente, debe también consultar dicho apartado.

EJEMPLO 3.20

```
\newlength{\AjAn}
\settowidth{\AjAn}{\input{ajedrez.pic}}
\addtolength{\AjAn}{1em}
\piccaptioninside
\pichskip{1.5em}
\piccaption{Los 64 escaques del ajedrez}
\parpic(1.6\AjAn,1.6\AjAn)[sr]{%
    \input{ajedrez.pic}}%
Se cuenta que el rey de un lejano país quedó enormemente entusiasmado con el juego del ajedrez y que para mostrar su gratitud al inventor le dijo que le pidiera lo que quisiera. El inventor reflexionó un momento y pidió al rey que le diera un grano de trigo por la primera casilla, dos por la segunda, por la tercera el doble de la anterior, es decir, cuatro, y así sucesivamente. Dícese que el rey se puso triste... le costaba la mitad de su reino.
```

Una figura marginada con el paquete `picins`; la leyenda, con la declaración `\piccaptioninside`, se ajusta siempre a la parte inferior de la caja



Se cuenta que el rey de un lejano país quedó enormemente entusiasmado con el juego del ajedrez y que para mostrar su gratitud al inventor le dijo que le pidiera lo que quisiera. El inventor reflexionó un momento y pidió al rey que le diera un grano de trigo por la primera casilla, dos por la segunda, por la tercera el doble de la anterior, es decir, cuatro, y así sucesivamente. Dícese que el rey se puso triste... le costaba la mitad de su reino.

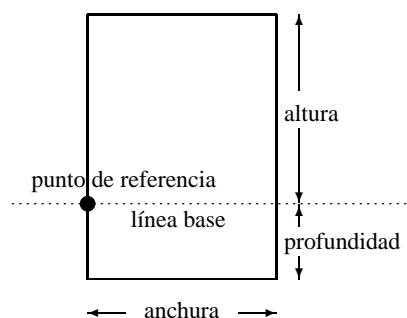
Figura 1: Los 64 escaques del ajedrez

3.5. Algo más sobre cajas

EN la lección 18 vimos algunos comandos que permitían crear y enmarcar cajas, en particular cajas capaces de contener cualquier cosa, llamados entornos `minipage`. En esta sección vamos a profundizar más: aprenderemos a manejar las cajas que `TEX` utiliza, que son más generales que las de `LATEX`. Veremos también cómo poner marco a una página entera y cómo sobreescribir el contenido de una caja con otra, en particular, describiremos un procedimiento para poner marcas al agua. Pero antes de nada, ¿qué es una caja? Realmente, ¡para `TEX`, todo es una caja!, o dicho de otro modo

T_EX sólo maneja cajas.

En efecto, cuando T_EX compone el documento no tiene ni idea de cuál es el aspecto de los símbolos como: s, ó, l, j, α , Σ , etc. Únicamente piensa en ellos como cajas con tres dimensiones, *anchura*, *altura* y *profundidad*, medidas todas con relación a un *punto de referencia*.

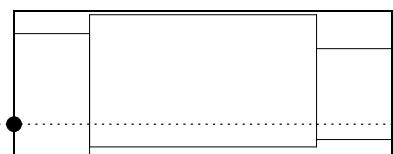


Cuando T_EX va realizando la composición de un párrafo, coloca los caracteres alineados, unos junto a otros, y todas las cajas de éstos tienen los puntos de referencia en una misma línea imaginaria que se llama *línea base*. Cada línea de texto se convierte, una vez finalizada, en una caja (con su anchura, altura y profundidad) y estas cajas, a su vez, pasan a ser elementos de otras cajas más grandes, y así sucesivamente hasta formar la caja de la página. Podría decirse que el trabajo de T_EX consiste únicamente en crear cajas y ponerlas unas junto a otras. Eso es lo que ha ocurrido en el rótulo que aparece centrado unas pocas líneas más arriba; en él pueden apreciarse las cajas de los caracteres y una caja que afecta al conjunto y que ha sido pintada un poco más grande de lo que es en realidad para facilitar su identificación.

A primera vista, el trabajo de T_EX resulta muy sencillo: sólo tiene que ir reuniendo cajas horizontalmente para formar líneas, las cuales se disponen verticalmente para formar los párrafos. Los distintos párrafos (o parte de ellos) se disponen también verticalmente para ir formando las páginas impresas que todos vemos, y así sucesivamente. La diferentes formas de poner las cajas unas junto a otras, ya sea una al lado de la otra en modo horizontal, o apilándolas en modo vertical, es lo que diferencia los modos de trabajo de T_EX; se comprende así que pueden diferenciarse dos modos de trabajo, el horizontal y el vertical. En realidad existe un tercer modo, el modo matemático, y, a su vez, cada uno de estos modos se subdivide en otros dos.

T_EX es un *cajista de múltiple personalidad*, de tal forma que según la personalidad que adopte en cada momento, irá disponiendo las cajas de una determinada manera. Más aún, el efecto de algunos comandos depende de la posición que ocupan en el documento fuente (en realidad, dependen de la «personalidad» de T_EX cuando los interpreta).

- **Modo horizontal.** T_EX agrupa las cajas alineándolas horizontalmente unas junto a otras, manteniendo los puntos de referencia sobre la línea base, creando una nueva caja que tendrá por anchura la suma de las anchuras de las cajas que la forman, y por altura y profundidad la mayor de las alturas y profundidades de las cajas, respectivamente. Pero T_EX puede trabajar en modo horizontal de dos maneras distintas:



- * Modo horizontal ordinario. Es la «personalidad» de T_EX cuando está construyendo los párrafos; va alineando horizontalmente todos los caracteres o cajas que hay en un párrafo y después va cortando la lista para crear las diferentes líneas, todas de la

misma anchura. En este trabajo, \TeX puede ir estirando o contrayendo los espacios (técnicamente, «glues») para conseguir el objetivo final.

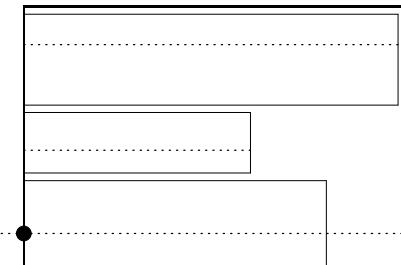
- * **Modo horizontal restringido.** Mientras \TeX trabaja en este modo, sólo puede alinear las cajas horizontalmente para formar una nueva caja, sin posibilidad de dividir la caja resultante en cajas más pequeñas. Por ejemplo, cuando esta «personalidad» es la dominante, \TeX no entiende los saltos de línea, de párrafo o de página. En general, \TeX no entiende todos aquellos comandos cuyo resultado sea poner los caracteres en un lugar distinto de la línea base actual.

- **Modo vertical.** \TeX agrupa las cajas verticalmente unas sobre otras, manteniendo los puntos de referencia en la misma vertical, creando una nueva caja que tiene por anchura la mayor de las anchuras de las cajas, y la suma de su altura y profundidad (*altura total*) coincide con la suma de todas las alturas y profundidades de las cajas que la forman. El punto de referencia suele ser el punto de referencia de la caja inferior, salvo que se modifique expresamente.

Pero al igual que ocurría en el modo horizontal, \TeX puede trabajar en modo vertical de dos formas diferentes:

- * **Modo vertical ordinario.** Es el modo por defecto y en él \TeX recoge las cajas, todas de la misma anchura, que ha realizado durante los modos horizontal y matemático y las va empaquetando verticalmente (siguiendo las diversas instrucciones que nosotros le queremos dar) formando una larga lista. Posteriormente, el procedimiento de salida ('the output routine') permite ir creando las páginas tal y como las vemos impresas, con sus cabeceras, sus pies de páginas, sus notas al margen, etc.
 - * **Modo vertical interno.** Es la «personalidad» dual del modo horizontal restringido, en el cual \TeX se limita a apilar verticalmente las cajas (manteniendo los puntos de referencia en la misma vertical) para crear una nueva caja indivisible. Por ejemplo, esto sucede cuando se construyen las columnas de una tabla de datos o de una matriz.
- **Modo matemático.** \TeX adopta este modo de trabajo cuando tiene que escribir un símbolo matemático o componer una fórmula matemática. Este modo tiene dos variantes:

- * **Modo matemático ordinario.** En este modo \TeX compone la fórmula matemática sabiendo que va a estar dentro de un párrafo, por lo que intenta que la altura total de la línea en la que esté no sea demasiado distinta a la del resto de líneas. Así, por ejemplo, utiliza operadores de un tamaño reducido, y los subíndices y superíndices están colocados de forma especial.
- * **Modo matemático resaltado.** Es la forma en que \TeX trabaja cuando queremos que la fórmula aparezca centrada y resaltada. Entonces se utilizan operadores de un tamaño mayor, y los espacios se manejan de forma totalmente distinta.



En el apartado I.16.1 señalamos las distintas declaraciones para entrar en modo matemático, tanto ordinario como resaltado. El modo matemático únicamente puede ser procesado

como material horizontal y por ello \TeX inicia un nuevo párrafo si, estando en modo vertical, encuentra un \$, o cualquier otra indicación que marque el comienzo del modo matemático.

3.5.1. El manejo de las cajas en \TeX

Las cajas que hemos considerado en la lección 18 son, en general, suficientes para dar respuesta a la mayor parte de las necesidades de un usuario medio e incluso avanzado. Pero conviene saber que hay otras cajas, las cajas de \TeX , más flexibles aún y sobre las cuales están construidas las cajas de \LaTeX . La mayor limitación de las cajas de \LaTeX es que no se dispone de una herramienta cómoda para apilar cajas en modo vertical; únicamente $\backslash\parbox$ y los entorno \minipage y \tabular prestan algún servicio en esa dirección pero resultan insuficientes. La sintaxis utilizada por estas cajas de \TeX es diferente de la de \LaTeX y, para evitar sorpresas desagradables, no deben mezclarse ambos tipos de cajas. En esta sección vamos a comentar brevemente algunos comandos de \TeX (no de \LaTeX) para construir y gestionar cajas, cuyo uso puede, en ocasiones, resultar muy práctico. Aunque sean comandos de \TeX se pueden utilizar también dentro de un documento escrito en \LaTeX .

Cajas horizontales: el comando $\backslash\hbox$

Cuando \TeX trabaja en modo horizontal ordinario va colocando las cajas (en particular las de los caracteres) de izquierda a derecha hasta encontrar el final del párrafo. Cuando llega al final, se detiene y, comenzando por el primer carácter, empieza a trocear esa larga caja horizontal en cajas más pequeñas, cuyas longitudes (salvo eventualmente la última) es aproximadamente la anchura de la línea, utilizando para ello la segmentación silábica y olvidando las elasticidades, si las hay.

Para construir una caja indivisible, de forma que el contenido se escriba de izquierda a derecha, \TeX proporciona el comando

$\backslash\hbox{[Material]}$

El argumento *Material* es procesado en modo horizontal restringido y puede estar formado por muchas cajas. Las dimensiones de la caja construida con un comando $\backslash\hbox$ son las siguientes: la altura y la profundidad son iguales, respectivamente, a la mayor de las alturas y profundidades de todas las cajas que la componen, mientras que la anchura es la suma de las anchuras de todas ellas, más las longitudes incluidas directamente, si las hay. Los comandos y entornos de \LaTeX utilizan internamente comandos $\backslash\hbox$ para realizar su trabajo.

Conviene hacer notar que el comando $\backslash\mbox$ de \LaTeX no es más que $\backslash\hbox$; en realidad, \LaTeX da la siguiente definición:

```
\def\mbox#1{\leavevmode\hbox{#1}}
```

donde el comando $\backslash\leavevmode$ significa salir del modo vertical, si eventualmente estamos en él.

El comando $\backslash\hbox$ (al igual que $\backslash\mbox$) no produce ningún recuadro. En los ejemplos que siguen el recuadro se muestra, únicamente, para delimitar el contorno de la caja.

Cada caja creada mediante $\backslash\hbox$ tiene una anchura natural que depende del *Material* incluido en ella. Pero, al igual que ocurre con el comando $\backslash\makebox$ (véase la lección 18), es posible fijar

de antemano dicha anchura. Existen dos formas de hacerlo y ambas utilizan una sintaxis diferente de la que es habitual en L^AT_EX.

<code>\hbox to Ancho{Material}</code>	<code>\hbox spread Ancho{Material}</code>
---------------------------------------	---

La primera forma crea una caja cuya anchura es igual a *Ancho* y en ella se coloca el *Material* de izquierda a derecha. Si la anchura del *Material* es menor que *Ancho*, además se estirarán los espacios elásticos para tratar de ocupar todo el espacio disponible; por el contrario, si la anchura del *Material* es mayor que *Ancho*, entonces parte del *Material* se sobreescibirá con el texto que venga a continuación de la caja. El compilador enviará un mensaje del tipo ‘Underfull ...’ en el primer caso y ‘Overfull ...’ en el segundo.

La segunda forma de modificar la anchura de una caja, mediante la partícula *spread*, sirve para aumentar (o disminuir cuando el valor es negativo) la anchura natural de la caja en una longitud igual a *Ancho*.

EJEMPLO 3.21

```
La \hbox to 4cm{conjetura de Goldbach}
es que todo número \hbox{par mayor que 2}
es suma \hbox to 18pt{de dos} números primos...
\hbox spread 4mm{La geometría global} o el
análisis han \hbox spread -4mm{surgido} del
cálculo \hbox spread 1cm{\hss infinitesimal\hss}.
```

La conjetura de Goldbach es que todo número par mayor que 2 es suma ~~de dos~~ números primos. Nadie ha conseguido demostrarlo.

La geometría global o el análisis han surgido del cálculo infinitesimal.

Lo habitual es utilizar esta forma de construir cajas horizontales junto con longitudes elásticas como `\hss`, `\hfil` y `\hfill` (véase la sección 1.5.2). Por ejemplo, algunas aplicaciones de estos comandos primitivos son los siguientes comandos de T_EX:

```
\def\leftline#1{\hbox to \hsize{#1\hss}}
\def\rightline#1{\hbox to \hsize{\hss#1}}
\def\centerline#1{\hbox to \hsize{\hss#1\hss}}
\def\rlap#1{\hbox to 0pt{#1\hss}}
\def\llap#1{\hbox to 0pt{\hss#1}}
```

donde `\def` se presenta en el capítulo 8. El comando `\hss` indica un espacio elástico que puede extenderse hasta el infinito y encogerse hasta el menos infinito, lo que lo distingue de los comandos `\hfil` y `\hfill` que varían entre cero e infinito. El funcionamiento de los comandos `\...line` ha sido descrito en el apartado I.3.3; el ejemplo 1.16 muestra el efecto de los comandos `\rlap` y `\llap`, el último de los cuales aparece también utilizado en los entornos `enumerate` e `itemize` para construir la etiqueta de los ítems mediante el comando `\makelabel` (véase la sección 3.2.2).

A modo de resumen, podemos decir que una caja `\hbox` contiene en su interior otras cajas, que aparecen organizadas horizontalmente de izquierda a derecha, en una única línea, formando un conjunto indivisible, a pesar de que la longitud total pueda superar el ancho de línea disponible. Si, además, la anchura se ha fijado de antemano, ésa será la anchura que T_EX reserve para ella, con independencia de que haya objetos que se salgan de dicha caja, produciendo, eventualmente, superposiciones con las cajas que van a continuación de ella. A veces, el comando se utiliza para

impedir que algunas palabras puedan ser separadas por \TeX en líneas diferentes, o para agrupar varias cajas en una sola.

Cajas verticales: el comando \vbox

Un segundo tipo de cajas son las cajas verticales, cuya principal característica es que el material que contienen es colocado verticalmente, de tal forma que los puntos de referencia estén alineados verticalmente. En cierto sentido este tipo de cajas es dual de las cajas horizontales construidas con \hbox , que se componían en modo horizontal restringido, mientras que este nuevo tipo de cajas se componen en modo vertical interno. Las cajas verticales se construyen mediante el comando:

$\boxed{\text{\vbox}\{Material\}}$

¿Cuáles son las dimensiones de una caja vertical? La anchura de una caja vertical es igual a la mayor de las anchuras de las cajas horizontales que la componen. Esta regla tiene dos excepciones: si la caja contiene texto en el nivel más alto (véase el ejemplo 3.22), la anchura es la anchura de una línea de texto (\hspace); si la caja contiene el comando \vrule , entonces la anchura también está determinada por \hsize (véase la página 322 para una descripción de \hrule y \vrule). La profundidad de una caja vertical es, esencialmente, igual a la profundidad de la última componente de la caja, aunque su valor está limitado por el valor de la constante \boxmaxdepth ; si la última componente no tiene dimensiones, entonces la profundidad es cero. La altura es la suma de las dimensiones verticales (altura y profundidad) de todas las cajas menos la profundidad de la última componente, más las longitudes incluidas directamente, si las hay.

El comando \vbox inicia el modo vertical (el modo vertical interno), pero \TeX puede estar en modo vertical sin necesidad de haber sido incitado a ello por un comando \vbox ; de hecho, ése es el modo en el que \TeX inicia la composición de un documento. En el modo vertical ordinario \TeX está construyendo la página como si fuera una caja vertical enorme, pero como la página tiene una altura fijada, cuando se alcanza dicha altura, \TeX archiva la caja de la página e inicia, en modo vertical ordinario, una nueva página (por eso es necesario, en ocasiones, utilizar un comando $\text{\mbox}\{\}$, a fin de hacerlo entrar en modo horizontal y que haga lo que esperamos; hay varios ejemplos a lo largo del libro, uno de ellos es el 1.16, en la página 200). Por el contrario, en el modo vertical interno, desde que se abre la llave hasta que se cierra va recibiendo cajas y apilándolas con sus puntos de referencia alineados verticalmente (con la salvedad de que lo primero que reciba sea texto sin meter en una caja horizontal), aunque se supere la altura de la página.

EJEMPLO 3.22

```
\vbox{Si contiene texto directo\... %
 \hbox{Caja primera}\hbox{Caja segunda}}
 \vbox{\hbox{Cuando no hay texto}%
 \hbox{Otra caja}\hbox{La última}}
 \vbox{\hsize 3cm Una variante de la primera...
 \hbox{Caja primera} \hbox{Caja segunda}}
```

Si contiene texto directo...	Caja primera	Caja segunda
Cuando no hay texto	Una variante de la	
Otra caja	primera...	Caja primera
La última		Caja segunda

Apilar cajas con \vbox ; observe que el comportamiento es diferente según que exista o no texto en el nivel más alto

De forma análoga a lo que ocurre con las cajas horizontales y `\hbox`, es posible fijar de antemano la altura de una caja vertical; podemos hacerlo de las dos formas siguientes:

`\vbox to Alto{Material}`

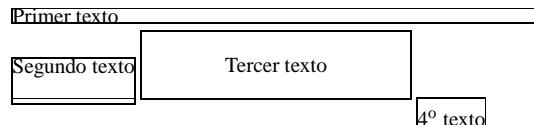
`\vbox spread Alto{Material}`

El significado de las variables es idéntico al descrito en el caso del comando `\hbox`.

La flexibilidad y comodidad de `\vbox` no puede obtenerse con ningún comando de L^AT_EX. Los entornos tabular realizan en las columnas de tipo 1, pero también hacen otras cosas y resultan menos «limpios» (véase el ejemplo 3.24). Para crear en L^AT_EX una caja de altura prefijada el único recurso es `\parbox`, pero utilizarlo requiere fijar también la anchura, lo cual, en muchos casos, es un inconveniente.

EJEMPLO 3.23

```
\parindent 0pt \vbox{Primer texto}\par
\vbox to 6mm{\hbox{Segundo texto}}
\vbox to 10mm{\hsize 4cm%
\vfil\centerline{Tercer texto}\vfil}
\vtop to 4mm{\vfil\hbox{4º texto}}
```



Observe que cuando la altura de la caja vertical supera a la que le correspondería por su contenido, el espacio sobrante se acumula en la parte inferior de la misma

Una caja `\vbox` puede aparecer en cualquier momento dentro de un párrafo. Su punto de referencia se alinearán horizontalmente con la línea base, lo que haya una gran separación con la línea que le precede o que le sigue. Si la caja está al final de una página es posible que se traslade al comienzo de la siguiente, provocando al compilar el mensaje de precaución ‘page underfull’.

EJEMPLO 3.24

```
\addtolength{\fboxsep}{3pt}
\ovalbox{\hbox to .9\hsize{%
\vbox{\hbox{«Caminante, son tus huellas}}%
\hbox{el camino, y nada más;}}%
\hfil\vbox{\hbox{caminante, no hay camino,}}%
\hbox{se hace camino al andar}}}}
```

«Caminante, son tus huellas caminante, no hay camino,
el camino, y nada más; se hace camino al andar»

«Caminante, son tus huellas caminante, no hay camino,
el camino, y nada más; se hace camino al andar»

Cajas horizontales y verticales anidadas. El código que figura a la izquierda es el responsable del primero de los óvalos que circunda el poema de Antonio Machado. El segundo, cuyo código no aparece, se ha construido con sendos entornos tabular de una columna separados entre sí por `\hfil` y se ha eliminado, además, el espacio correspondiente a los separadores de columnas

`\vtop to Alto{Material}`

`\vcenter to Alto{Material}`

Estos comandos funcionan de forma análoga a `\vbox`. La línea base de una caja obtenida con `\vbox` coincide con la de la última caja incluida en la `\vbox`. Por el contrario, la línea base de una caja `\vtop` coincide con la de la primera caja incluida en la `\vbox`. Finalmente, la línea base de la caja creada con el comando `\vcenter` la divide en dos mitades iguales; este comando sólo puede ser empleado en modo matemático.

Desplazamiento vertical de cajas

Una caja puede moverse en los cuatro sentidos: arriba, abajo, derecha o izquierda. Si la caja es horizontal (ha sido creada mediante un comando `\hbox`) entonces sólo puede desplazarse verticalmente, a no ser que se encuentre dentro de una caja `\vbox`, en cuyo caso también puede moverse horizontalmente.

Los comandos para desplazar una caja verticalmente son:

<code>\raise Desplazamiento</code>	<code>\lower Desplazamiento</code>
------------------------------------	------------------------------------

donde *Desplazamiento* indica cualquier longitud. Funcionan únicamente en modo horizontal. Los comandos `\raise` y `\lower` son esencialmente el mismo, ya que $\raise \text{Desplazamiento} = \lower -\text{Desplazamiento}$. Cuando se usan estos comandos, la línea base permanece en el mismo lugar que antes de proceder al desplazamiento. La nueva altura y profundidad están determinadas por la máxima distancia entre la línea base y la parte superior e inferior de la caja una vez que ha sido desplazada. La altura y la profundidad de una caja `\hbox` deben ser positivas; la anchura, sin embargo, puede ser no positiva, en cuyo caso el texto que sigue se imprimirá encima. Uno de los ejemplos más típicos en los que se utiliza este tipo de desplazamiento es, precisamente, en el logotipo TEX, cuya definición es la siguiente:

```
\hbox{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125ex X}
```

El comando

<code>\kern Longitud</code>

indica desplazamiento. Si se utiliza en una caja `\hbox` se refiere a desplazamiento horizontal, mientras que dentro de una caja `\vbox` indica desplazamiento vertical. El valor de *Longitud* puede ser negativo.

EJEMPLO 3.25

El desplazamiento vertical de `\raise5pt\hbox{cajas}` es `\lower-5pt\hbox{sencillo.}`
Se pueden desplazar cajas `\raise-5pt\hbox{horizontales}` así como cajas `\lower-5pt\vbox{\hbox{verti-}\hbox{cales}}`.

El desplazamiento vertical de `cajas` es `sencillo`. Se pueden desplazar cajas `horizontales` así como cajas `verti-cales`.

Desplazando verticalmente cajas

Desplazamiento horizontal de cajas

Los comandos para desplazar una caja horizontalmente son:

<code>\moveleft Desplazamiento</code>	<code>\moveright Desplazamiento</code>
---------------------------------------	--

donde *Desplazamiento* indica cualquier longitud. Se utilizan para mover horizontalmente las componentes almacenadas en una caja vertical `\vbox`. Si el movimiento es hacia la izquierda, entonces la anchura de la caja no se ve afectada, cosa que sí puede ocurrir si el desplazamiento es a la derecha. La anchura de una caja vertical, una vez que alguna de sus componentes ha sido desplazada a

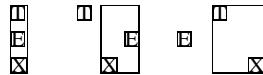
la derecha, se calcula comenzando en el punto de referencia y moviéndose hacia la derecha hasta encontrar la parte derecha de la componente más alejada.

EJEMPLO 3.26

```

\begin{center}
\vbox{\hbox{T}\hbox{E}\hbox{X}}\hspace{1cm}
\vbox{\moveleft10pt\hbox{T}%
\moveright10pt\hbox{E}\hbox{X}}\hspace{1cm}
\vbox{\hbox{T}\moveleft15pt\hbox{E}%
\moveright15pt\hbox{X}} \end{center}

```



Desplazando horizontalmente las cajas

Rayas horizontales y verticales: cajas negras

En T_EX hay dos tipos de cajas negras: las horizontales, `\hrule` (que son las que internamente usa el comando `\rule` explicado en la lección 18), y las verticales, `\vrule`. Ambas tienen tres dimensiones: anchura, altura y profundidad.

```
\hrule height Altura width Anchura depth Profundidad  
\vrule height Altura width Anchura depth Profundidad
```

No es necesario dar las tres longitudes, *Altura*, *Anchura* y *Profundidad*, para definir una raya; de hecho, pueden eludirse las tres, y si alguna de ellas se omite TeX le asignará un valor por defecto. El orden es irrelevante y alguna de las longitudes puede ser cero. Podría pensarse que los resultados obtenidos utilizando \hrule y \vrule son idénticos. Sin embargo, esto no es así, ya que TeX considera que \hrule es material vertical y, por tanto, puede ser parte de una lista vertical, mientras que \vrule es material horizontal y sólo puede aparecer en listas horizontales. En resumen, \vrule puede utilizarse dentro de un párrafo, como la raya vertical siguiente: | (que ha sido producida con \vrule width 0,4pt), o dentro de una caja \hbox. Por el contrario, \hrule debe usarse entre párrafos o dentro de una caja vertical \vbox.

EJEMPLO 3.27

Los comandos \texttt{\textbackslash hrule}... el comando \texttt{\textbackslash hrule} crea \vbox{\hrule height5pt width15pt depth0pt}, mientras que el comando \texttt{\textbackslash vrule} crea \vrule height5pt width15pt depth0pt.

Los comandos `\hrule` y `\vrule` son diferentes pero pueden generar rayas idénticas. Así, para `Altura=5pt`, `Anchura=15pt` y `Profundidad=0pt`, el comando `\hrule` crea , mientras que el comando `\vrule` crea .

Para convencerte de que son diferentes compile el código de la izquierda, tras eliminar la caja \vbox. Si ha entendido la discusión sobre los modos de trabajo podrá prever el resultado.

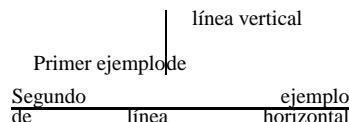
Dibujando rayas horizontales y verticales

Si se omite alguna de las medidas de la raya, TeX le asignará las siguientes:

- Altura 0,4 pt y profundidad 0 pt, si la raya es horizontal (`\hrule`).
 - Anchura 0,4 pt, si la raya es vertical (`\vrule`).
 - Las demás dimensiones de la raya las obtiene TEX extendiéndola indefinidamente hasta donde le permita el tamaño de la caja más pequeña que la contiene.

EJEMPLO 3.28

```
\hbox{Primer ejemplo\vrule de %
      \vbox to 25pt{\hbox{línea vertical}}}
\ vbox{\hbox to50mm{Segundo ejemplo}\hrule%
      \hbox to50mm{de línea horizontal}}
```



Otros comandos interesantes

En la sección 1.5.2 hemos descrito los comandos `\setto...` que permiten asignar a una longitud existente alguno de los valores de las dimensiones de una caja. Las dimensiones de una caja pueden obtenerse también con los comandos

`\wd\NombreCaja` `\ht\NombreCaja` `\dp\NombreCaja`

que proporcionan, respectivamente, la anchura, altura y profundidad de `\NombreCaja`. Hemos visto en la sección I.18.4 cómo guardar y reutilizar cajas en L^AT_EX. El comando `\sbox` guarda las cajas en formato de cajas horizontales `\hbox`. Hay un comando más general de T_EX

que permite guardar cajas, tanto

donde `\NombreCaja` es una caja declarada y `Caja` es cualquier caja horizontal o vertical. En la asignación anterior el signo `-` puede claudicar.

TEX dispone de 256 registros para guardar cajas, que identifica mediante un número, `\box0` a `\box255`; la caja `\box255` está reservada para la caja de la página. No es recomendable que el usuario utilice tales números de forma directa, sino a través del comando `\setbox` que asigna internamente a la nueva caja `\NombreCaja` un número que permite identificarla.

\box\NombreCaja

\copy\NombreGaia

Son dos comandos que permiten insertar, en un lugar concreto del documento, el contenido de una caja previamente guardada. La diferencia entre ambos es que el primero, al ser usado, borra el contenido de la caja, mientras que el segundo usa el contenido de la caja sin borrarlo.

EJEMPLO 3.29

```
\newsavebox{\MiCaja}\sbox{\MiCaja}{;Hola!}
\usebox{\MiCaja} ¿Has dicho \usebox{\MiCaja}?
Sí, he dicho \copy{\MiCaja} A que ya no puedes
decir otra vez \box{\MiCaja} ¡Anda! ya no puedo
decir \usebox{\MiCaja}
```

¡Hola! ¿Has dicho ¡Hola!? Sí, he dicho ¡Hola! A que ya no puedes decir otra vez ¡Hola! ¡Anda! ya no puedo decir

Los comandos que siguen sirven también para reutilizar cajas pero «desmembrándolas» en sus componentes en el momento de ser usadas.

```
\unhbox\NombreCaja  
\unhcopy\NombreCaja
```

Los comandos de la primera columna abren el contenido de la caja `\NombreCaja`, que debe ser una caja horizontal, y lo añaden al material horizontal que está componiéndose. Por tanto, estos comandos deben insertarse en modo horizontal, o en caso contrario iniciarán un nuevo párrafo. El primer comando usa el contenido y vacía la caja después de usarlo, mientras que el segundo no la vacía. Los comandos de la segunda columna son análogos, pero aplicados a cajas verticales y material vertical. Estos comandos deben insertarse en modo vertical o, en caso contrario, finalizarán el párrafo en curso.

`\lastbox`

Este comando permite recuperar el «último ítem» de una lista de cajas. Si el último ítem de la lista que estamos procesando en modo horizontal o en modo vertical interno es un `\hbox`, `\vbox` o `\vtop`, el comando `\lastbox` recupera el último ítem y lo elimina de la lista. El comando no puede usarse en los modos vertical ordinario y matemático.

<code>\lineskip</code>	<code>\nointerlineskip</code>
<code>\lineskiplimit</code>	<code>\offinterlineskip</code>

Cuando una caja se incorpora a una lista vertical de cajas, \TeX añade un espacio elástico para que la separación entre la línea base de la nueva caja y la línea base de la caja anterior sea exactamente igual a `\baselineskip`. Pero si la interlínea calculada mediante esta regla hace que la distancia entre la parte superior de la nueva caja y la parte inferior de la caja anterior sea inferior a `\lineskiplimit`, entonces se utiliza `\lineskip` como separación elástica entre las cajas. En otras palabras, cuando se apilan cajas verticalmente, la separación entre dos cajas consecutivas será `\baselineskip`, excepto si las dos cajas se quedan demasiado juntas, en cuyo caso se utiliza la longitud elástica `\lineskip`.

Si no deseamos que \TeX inserte espacio elástico entre las cajas, entonces debemos utilizar `\nointerlineskip`. En cuanto a `\offinterlineskip`, significa que no habrá espacios en blanco entre las líneas, de modo que \TeX no insertará ningún espacio adicional entre las diferentes cajas (este comando es más drástico que `\nointerlineskip`). Es equivalente a

```
\baselineskip-1000pt
\lineskip0pt
\lineskiplimit\maxdimen
```

La tradición española para escribir un verso que necesita dos líneas es diferente de la que \LaTeX realiza con el entorno `verse`. En nuestra tradición la segunda línea aparece justificada a derecha y se inicia con el carácter [, tal y como aparece en el ejemplo 3.30.

Finalizaremos esta sección presentando una aplicación de los comandos anteriores a la construcción un entorno `poema` acorde con nuestras tradiciones.

 El código que sigue resuelve este problema. Puede ser incorporado a un fichero personal de estilo (lo que ayuda a mantener un preámbulo más «ordenado») o bien ser incluido en el preámbulo del documento, lo que resulta una solución más satisfactoria, especialmente si se van a compartir con otras personas archivos que utilicen el entorno `poema`. En esta segunda hipótesis, obviamente, es necesario rodear la definición del entorno con los comandos `\makeatletter` y `\makeatother`.

```
\newenvironment{poema}
{\def\\{\egroup\ver{\unhbox\z@}\setbox\z@\hbox\bgroup
 \ignorespaces}
 \def\fin{\egroup\ver{\unhbox\z@}\bigskip\setbox\z@\hbox\bgroup
 \ignorespaces}
 \def\ver##1{\item\relax
 \setbox\z@\hbox{##1}
 \ifdim\wd\z@>\linewidth
 \setbox@tempboxa\vbox{\hsize\linewidth\unhcopy\z@}
 \unvbox@tempboxa
 \setbox@ne\lastbox
 \setbox@tw@\lastbox
 \vbox{\nointerlineskip\offinterlineskip
 \hbox to\linewidth{\unhbox@tw@}
 \hbox to\linewidth{\null\hfill[\unhbox@ne]}
 \else\unhcopy\z@\fi }
 \trivlist\itemsep8pt\setbox\z@\hbox\bgroup\ignorespaces}
 {\egroup\ver{\unhbox\z@}\endtrivlist}
```

Este código no es sencillo, y merece algunos comentarios adicionales. Los comandos `\ifdim`, `\fi`, `\bgroup`, `\egroup` y `\def` se describen en el capítulo 8; `\ignorespaces` está descrito en el apartado PARA SABER MÁS del capítulo 1; y, finalmente, nos quedan los comandos

`\z@ \ne \tw@ \tempboxa`

que corresponden a cajas ya definidas en L^AT_EX. La motivación para el empleo de estos últimos es «ecológica»: también los recursos de T_EX son limitados... ¿Por qué crear nuevos registros para almacenar cajas (que es para lo que se utilizan dichos comandos en el entorno `poema`) si podemos reutilizar registros ya existentes? Reciclar requiere tener los conocimientos oportunos.

Una aplicación del entorno `poema` se muestra en el siguiente ejemplo.

EJEMPLO 3.30

```
\begin{poema}
Recuerdo que la profundidad de tu audacia
creadora\\
fue compañera de los hombres que de ti se
fiaron,\\
mientras que los que obraron de espaldas a
tus designios\\
construyeron la trampa que arruinaría su
propio esfuerzo.
\end{poema}
```

<pre>Recuerdo que la profundidad de tu audacia [creadora fue compañera de los hombres que de ti se [fiaron, mientras que los que obraron de espaldas a [tus designios construyeron la trampa que arruinaría su pro- [pio esfuerzo.</pre>
--

Cada verso acaba con `\` y cada estrofa del poema se finaliza con el comando `\fin`. Versos extraídos de «Canciones del hombre nuevo (1983–85)», Antonio López Baeza

3.5.2. Más posibilidades para enmarcar y sobreescribir cajas

En el apartado I.18.1 describimos `\shadowbox`, `\doublebox`, `\ovalbox` y `\ovalbox`, comandos del paquete `fancybox`, destinados a crear cajas con marcos llamativos. En esta sección nos ocuparemos de otros comandos implementados en dicho paquete que permiten aislar la caja de la página entera, con cabecera y pie, o sin ellos, para tratarla como argumentos de los comandos anteriores y otros similares.

También veremos comandos que permiten sobreescribir el contenido de una caja con otro material; en particular mostraremos un modo mediante el cual puede sobreescribirse la caja de la página para incluir «filigranas» o «marcas al agua».

```
\fancypage{Comandos1}{Comandos2}
\thisfancypage{Comandos1}{Comandos2}
```

El primero de ellos es una declaración con efectos a partir de ese punto. El segundo es una declaración que afecta únicamente a la página en que aparece el comando.

Comandos1 Debe ser un comando cuyo argumento pueda ser una caja, o bien una colección de comandos, que finalicen con un comando cuyo argumento pueda ser una caja. Operan sobre la caja de la página antes de que se añada la cabecera y el pie de la misma. Se pueden incluir comandos como, por ejemplo, `\shadowbox`, precedidos de declaraciones como `\fboxsep` o `\shadowsize`.

Comandos2 Sobre la caja creada uniendo cabecera y pie a la creada por *Comandos1*, y con la misma sintaxis que aquéllos, operan ahora *Comandos2*.

EJEMPLO 3.31

```
% Enmarcado de la página:
% A) con \ovalbox el cuerpo
%   modificando previamente el valor de
%   \fboxsep (véase la lección 18);
% B) con \fbox la página completa incluyendo
%   caceras y pies.
\fancypage{\fboxsep 8pt\ovalbox}{\fbox}

% Marca al agua o filigrana en color gris
% del 70% (véase la sección sobre colores
% en este mismo capítulo y lo relativo
% a rotación y redimensionado de cajas
% en el capítulo de gráficos)
\fancyput(.5in,-8.5in){%
  \color[gray]{.7}\rotatebox{50}{%
    \resizebox{7in}{!}{\Huge 31-05-2003}}}
\include{lección}
```

Páginas enmarcadas y marcas al agua; el color y la rotación requieren programas que soporten estas posibilidades

Llamamos la atención sobre el hecho de que, para conseguir el resultado buscado, el paquete modifica la «output routine» de L^AT_EX y puede producir resultados inesperados con clases de documento o paquetes que también actúen sobre ella.

Se dispone también de comandos para poner objetos arbitrarios en cualquier posición de la página, permitiendo, por ejemplo, poner una «marca al agua» en las páginas de un documento.

<code>\fancyput(x,y){Material}</code>	<code>\fancyput*(x,y){Material}</code>
<code>\thisfancyput(x,y){Material}</code>	<code>\thisfancyput*(x,y){Material}</code>

El argumento (x,y) es opcional y sirve para definir la posición en que se va a colocar el *Material* con relación a un sistema de coordenadas cuyo origen $(0pt,0pt)$ se sitúa una pulgada hacia abajo y otra hacia la derecha de la esquina superior izquierda del papel. Si este argumento opcional no se utiliza, el *Material* se ubica en el origen de coordenadas. Estos comandos son parecidos al comando `\rput` del entorno `pspicture` (véase la sección 4.4) salvo que, en dicho entorno, la unidad de longitud está fijada de antemano y las coordenadas pueden ser únicamente números, mientras que aquí debe especificarse la unidad de ambas coordenadas.

El comando `\thisfancyput` afecta únicamente a la página actual, mientras que el comando `\fancyput` es una declaración que afecta también a las páginas posteriores. Cada nueva utilización de uno de estos comandos reemplaza la definición anterior, salvo que se utilice la versión con asterisco `*`, en cuyo caso actúa de forma acumulativa, es decir, imprime todo el *Material* que contengan los diferentes `\fancyput` o `\thisfancyput`.

Los comandos `\fancyput` y `\thisfancyput` sobreimprimen la caja de la página con *Material* colocado en cierta posición. Pero se dispone también de comandos para sobreimprimir cualquier caja, no necesariamente la caja de la página.

Los comandos

<code>\boxput(x,y){Material1}{Material2}</code>	<code>\boxput*(x,y){Material1}{Material2}</code>
---	--

permiten sobreimprimir la caja del *Material2* con *Material1*. Como en el caso de los comandos `\fancyput` y `\thisfancyput`, los parámetros (x,y) que determinan las coordenadas en las que se coloca el *Material1* son optativos y su valor por defecto es $(0,0)$.

Sin embargo existen diferencias de sintaxis entre estos nuevos comandos y los comandos descritos anteriormente, tanto en el sistema de coordenadas como en el funcionamiento de la versión con asterisco `*`. Diferencias que pasamos a describir.

- Con relación al sistema de coordenadas la primera diferencia estriba en que el origen de coordenadas se sitúa en el centro de la caja del *Material2*, en lugar de hacerlo en la esquina superior izquierda de dicha caja. La segunda diferencia es que las coordenadas no tienen unidad, son sólo números (decimales) entre -1 y 1 . Lo que significa que las coordenadas son relativas a la caja del *Material2*. Así, por ejemplo, la esquina superior izquierda tiene coordenadas $(-1,1)$ y las de la esquina inferior derecha son $(1,-1)$.
- La sobreimpresión del *Material1* puede hacerse colocándolo encima del *Material2*, tapando parcialmente a éste (comando con asterisco), o bien colocándolo debajo, siendo parcialmente tapado por éste (comando sin asterisco). La distinción entre ambos comportamientos sólo resulta visible con la utilización de colores transparentes, pero con tinta negra opaca no se

percibe diferencia alguna. La opción del comando sin asterisco es la más frecuente y es la utilizada en el caso de los comandos `\fancyput` y `\thisfancyput`.

El paquete `fancybox` implementa también entornos específicos para enmarcar listas, texto centrado, ecuaciones matemáticas, etc. Evidentemente, pueden conseguirse resultados parecidos a los que el paquete proporciona incluyendo los entornos `center`, `itemize...` como argumentos de un entorno `minipage`, pero la utilización de los entornos `Bcenter`, `Bitemize`, etc., resulta muy conveniente. Listamos a continuación entornos definidos en el paquete, cuyo nombre hace referencia a entornos ya descritos en la primera parte y a otros que aparecen en esta segunda parte del libro, a los que antepone el prefijo ‘B’ y cuyo funcionamiento es análogo al del correspondiente entorno sin ‘B’.

<code>Bcenter</code> [Posición]	<code>Bitemize</code> [Posición]
<code>Bflushleft</code> [Posición]	<code>Benumerate</code> [Posición]
<code>Bflushright</code> [Posición]	<code>Bflushright</code> [Posición]
	<code>Bdescription</code> [Posición]
	<code>Blist</code> [Posición]
	<code>Beqnarray</code>
	<code>Beqnarray*</code>

El argumento optativo *Posición* toma uno de los valores `t` o `b` cuyo significado es, como siempre, alineado por la parte superior o inferior de la caja. Para enmarcar estos entornos debemos incluirlos en el argumento de un comando `\fbox`, `\doublebox...`

EJEMPLO 3.32

```
\fboxsep5pt
\fbox{\begin{Beqnarray} e^{2\pi i} + 1 &= 0 \\ \sum_{n=1}^{\infty} \frac{1}{n^2} &= \frac{\pi^2}{6} \end{Beqnarray}}
```

$$\begin{aligned} e^{2\pi i} + 1 &= 0 & (3.1) \\ \sum_{n=1}^{\infty} \frac{1}{n^2} &= \frac{\pi^2}{6} & (3.2) \end{aligned}$$



PARA SABER MÁS

- El paquete `dcolumn` facilita la alineación vertical del separador decimal (o cualquier otro separador) en tablas. Permite evitar el uso manual del especificador `@`, lo que resulta muy útil en tablas grandes, sobre todo si son leídas de ficheros de datos. Consulte el archivo `dcolumn.pdf`.
- En el archivo `tabularx.pdf` encontrará la descripción de las herramientas del paquete `tabularx` que permiten construir tablas con columnas de igual anchura, mejorando el aspecto de las mismas.
- El paquete `float`, descrito en el archivo `float.pdf`, le permite crear nuevas familias de entornos flotantes. Podrá así clasificar los objetos grandes, los adecuados para poseer el carácter flotante, en más de dos clases y crear índices para todos ellos.

-  Cuando una figura se compone de varios objetos separados se necesita, en ocasiones, referirse a cada uno de ellos independientemente. El paquete `subfigure`, descrito en el archivo `subfigure.pdf`, proporciona los comandos adecuados.
-  El paquete `picins` sólo puede colocar objetos marginados al comienzo de los párrafos. El paquete `picinpar` puede conseguir colocarlos en cualquier esquina de un párrafo e, incluso, en el centro del mismo. Aunque esto tiene inconvenientes, y probablemente es de poca utilidad, consulte el archivo `picinpar.pdf` para aprender cómo hacerlo.
-  En el archivo `pies.pdf` puede encontrar información sobre mejoras de las notas a pie de página: cómo cambiar su numeración y formato, así como la descripción del paquete `endnotes` que posibilita la ubicación de todas las notas al final de cada capítulo o al final del documento.

Capítulo

4

Gráficos

ESTIMADO lector, ¿necesita incluir un gráfico en su documento? Si la respuesta es afirmativa nuestra primera recomendación es que lo tome con paciencia, aunque, a decir verdad, este consejo vale para cualquier intento de trabajar con L^AT_EX. ¿Ha conseguido ya alguna vez incluir gráficos en un documento L^AT_EX? ¿Ha leído, y recuerda, la lección 9? Resumamos su contenido.

Los gráficos electrónicos fueron clasificados según dos conceptos:

- *su procedencia*, distinguiendo aquellos generados por un programa externo (gráficos cerrados para L^AT_EX) y los realizados desde L^AT_EX, ya sea L^AT_EX básico o algún paquete;
- *su contenido*, distinguiendo entre gráficos «artísticos» y «lineales» o con estructura interna.

Lo esencial de esta clasificación es, en primer lugar, que existe un cierto paralelismo entre los tipos de cada una de las dos categorías: cerrados-artísticos y lineales-L^AT_EXeados. En segundo lugar, si los gráficos deben contener texto, entonces los generados por L^AT_EX y compañía son siempre preferibles, ya que podrán manejar los mismos tipos que el texto principal del documento. Puede que algunas personas consideren que no es grave tener, por ejemplo, una letra, que representa un cierto elemento, escrita de forma diferente en el gráfico y en el texto; para nosotros, que buscamos la perfección tipográfica en nuestros documentos, y de la que esperamos no haber quedado demasiado lejos en este libro, es de capital importancia. ¿De qué lado se encuentra usted?

La parte técnica de la lección 9 se refería exclusivamente a la inclusión de gráficos externos o «cerrados», destacando, por encima de todo, la idea de que los formatos gráficos electrónicos incorporables dependen del visor o impresor de documentos que manejemos. Un breve apunte sobre esta dependencia puede ser de interés para aquellos que gusten de conocer los entresijos de T_EX; desde luego, no para los que busquen sólo el lado pragmático. Como ya comentábamos en la lección 9, Donald Knuth dejó una puerta abierta para la adaptación de T_EX a las mejoras tecnológicas: se trata de un comando muy especial, el comando

```
\special{Acción}
```

Es muy especial porque ¡no hace nada!, o casi nada; se limita a incluir la *Acción* en el fichero DVI o PDF que resulta de la compilación. Por tanto, la filosofía es dejar al programa visor o impresor de dicho fichero la tarea de ejecutar la *Acción* (véase el esquema de funcionamiento en la página 6). En este caso la *Acción* es la inclusión de un fichero gráfico y, como se deduce de lo antes

afirmado, debe ser una sentencia en algún «lenguaje» que el programa en cuestión comprenda o sepa interpretar. Esto es lo que hace el paquete `graphicx` con la opción de controlador adecuada: adaptar la *Acción* a dicho controlador (programa).

Este capítulo se ajusta a las citadas clasificaciones de gráficos: las secciones segunda y tercera se ocupan de completar la información sobre la inclusión de gráficos externos, mientras que la última sección presenta `PSTricks`, la mejor de las herramientas de la familia `LATEX`, para la creación de gráficos lineales (y no tan «lineales»). El apartado PARA SABER MÁS, al final del capítulo, le mostrará otras posibilidades. Pero antes de todo ello, presentamos, en la primera sección, un pequeño algoritmo que, esperamos, le resulte útil cuando se enfrente a la necesidad de incluir gráficos en sus documentos.

4.1. Esbozo de una estrategia para la inclusión de gráficos

VISTAS las limitaciones en el formato de salida y en el formato de los gráficos, así como en su contenido, quizás se encuentre perdido, necesita incluir un gráfico, pero no sabe ni siquiera por dónde empezar... Procedamos a proponer un *esbozo de estrategia* que pueda allanar su camino.

Paso 1 Decidir el formato de salida final del documento `LATEX`. En esta decisión influye de forma esencial cuál deseamos que sea el destino final del documento: ¿será un documento electrónico para ser publicado en Internet o para una presentación?, ¿sólo queremos un texto impreso? En el segundo caso, impreso, no hay limitación alguna, cualquier formato de salida es bueno, las herramientas de conversión (resumidas al final de esta sección) nos permitirán la impresión desde cualquier formato habitual con `LATEX`: DVI, PS o PDF. Si se desea un documento electrónico la libertad se reduce. En principio sólo tendremos restricciones si deseamos una salida PDF que controle la forma de visualizar el documento (marcadores, miniaturas, transiciones de página) y/o incorpore anotaciones de cualquier tipo (texto, audio o vídeo, véanse los apartados 6.2.1 y 6.2.3). En estos casos nuestros gráficos tendrán que ser necesariamente adaptados a dicha salida. Existe todavía, sin embargo, una posibilidad para evitar esta restricción, que consiste en utilizar el programa comercial ACROBAT para convertir una salida PS en PDF, incorporándole los elementos de control y anotaciones; sin embargo, esta solución tiene el grave inconveniente de ser totalmente «manual» y no mecánica, ya que cualquier cambio en el documento fuente `LATEX` producirá un nuevo PS al que habrá que adjuntar, desde cero, todas las anotaciones(además de la cuestión comercial, por supuesto).

Paso 2 ¿Posee ya el gráfico en formato electrónico? Si la respuesta es afirmativa y el formato electrónico se adapta a la salida definida en el *paso 1* el proceso ha terminado, en lo que se refiere a «este gráfico»; vaya al *paso 3*. Para conocer si el formato se adapta o no a la salida, consulte el apartado I.9.2 o [10]. Si, en cambio, el formato electrónico no es de los incorporables a la salida por la que se ha optado, pase al proceso de *conversión de formatos gráficos*. Si, finalmente, la respuesta es negativa, vaya al proceso de *generación de gráficos*. Aun cuando la respuesta sea positiva, si el gráfico contiene texto (etiquetas, símbolos, acotaciones, etc.) plantéese la posibilidad de generar de nuevo el gráfico desde una herramienta de la familia `LATEX`, no tema caer en el perfeccionismo, sus esfuerzos se verán compensados.

Paso 3 ¿Necesita incluir más gráficos? Si la respuesta es negativa, finalizó la tarea para este documento. Si la respuesta es afirmativa, vuelva al *paso 2*.

Proceso de conversión de formatos gráficos Si ha llegado hasta este proceso es que necesita convertir un gráfico a un formato electrónico distinto del que ya posee. Algunas herramientas de conversión de formatos gráficos se describen en el apartado siguiente. Una vez elegida la que se adapta a su necesidad y utilizada para la conversión del gráfico, diríjase al *paso 3*.

Proceso de generación de gráficos Necesita crear un gráfico; ¿es de tipo lineal o artístico?

- Si es de tipo artístico necesitará cualquiera de las múltiples herramientas informáticas de diseño (más o menos complicada según sea nuestra imagen) o cualquier otra forma de generar el gráfico (escaneo a partir de una imagen, cámara digital, generación gráfica a través de cálculo numérico mediante alguna herramienta matemática, etc.). Siempre, ante varias posibilidades, conviene elegir aquella vía que nos permita obtener un fichero en un formato electrónico incorporable a la salida decidida en el *paso 1*.
- Si es de tipo lineal, no lo dude, elija las herramientas de la familia L^AT_EX. En la sección 4.4 se presentan algunos elementos básicos de PSTRicks, uno de los paquetes más potentes para generar gráficos, aunque requiere el uso de DVIPS, es decir, la salida es necesariamente PostScript. Hay muchas otras herramientas orientadas a este fin (véase el apartado PARA SABER MÁS al final de este capítulo). Nosotros hemos destacado el paquete PSTRicks, por encima de todas, tanto por sus posibilidades como por su sintaxis, muy similar a la de L^AT_EX. Su principal inconveniente es el de restringir la salida, que sólo puede ser PostScript. Una forma de cortocircuitar esta restricción consiste en construir cada gráfico con PSTRicks en un documento aparte, generar un fichero de salida con formato eps y, posteriormente, convertir éste a un formato incorporable e incluirlo en el documento. Esto ralentiza aún más la creación de gráficos, pero nos da toda la libertad de PSTRicks. Otras posibilidades, como el uso del paquete pdftricks, son brevemente comentadas en el apartado 4.1.1.

Una vez creado su gráfico, diríjase al *paso 3*.

4.1.1. Algunos métodos de conversión de formatos gráficos

En la estrategia anterior hemos podido comprobar que, en lo que respecta a la inclusión de gráficos en nuestros documentos L^AT_EX, antes o después, tendremos que recurrir a la conversión de un formato gráfico electrónico a otro formato distinto. A continuación citamos algunas de las herramientas disponibles para este objetivo¹. Para ser concretos nos ceñiremos a las conversiones que hagan posible obtener una salida PS o PDF.

En realidad el proceso de conversión se puede realizar a dos niveles:

1. Convertir los gráficos necesarios para poder obtener la salida deseada.
2. Obtener una salida cualquiera, que posiblemente necesite menos conversiones de gráficos, para después convertirla al formato de salida realmente deseado. Este procedimiento no es aconsejable y, desde luego, no permite incorporar anotaciones a una salida PDF.

¹La mayor parte de ellas se incluyen en el CD-ROM que acompaña al libro.

A continuación listamos algunos procedimientos de conversión.

De PS a PDF Ya se comentaba en el apartado I.1.4 que puede realizarse con el ejecutable PS2PDF, distribuido con GHOSTSCRIPT, o desde el menú Archivo|Convertir, opción pdfwrite, de GSVIEW.

De PS a EPS Esta conversión es esencial para poder incluir gráficos de formato PostScript en documentos L^AT_EX. En el ejercicio I.9.3 se explicaba la diferencia entre archivos PS y EPS. Puede obtener esta conversión mediante el menú Archivo|PS a EPS de GSVIEW que, además, le permite optar por un cálculo automático de la BoundingBox o por una determinación manual de la misma. Para poder realizar esta conversión, el fichero PS de partida sólo puede contener una página; en caso de que no sea así, utilice el menú Archivo|Extraer de GSVIEW para extraer una página de un PS y guardarla en otro archivo.

De EPS a PDF Una conversión muy útil para poder transformar los PostScript encapsulados en gráficos PDF, que pueden ser incorporados a un documento que se compilará con PDFL^AT_EX; por ejemplo, ésta es una buena solución para poder recurrir a PSTricks con el objetivo de crear gráficos que, mediando la conversión, se incluirán en un documento PDF. Puede recurrir al ejecutable EPSTOPDF, disponible en distintas plataformas, o a EPS2PDF, disponible para MS-WINDOWS, que le permite, con una interfaz cómoda, convertir varios archivos a la vez. Ambos necesitan del concurso de GHOSTSCRIPT.

De otros formatos (JPG, GIF, TIF, etc.) a EPS La variedad en este caso es enorme. En principio cualquier herramienta gráfica de la que disponga puede realizar esta tarea, por ejemplo FREE HAND, PHOTO SHOP, IMAGEMAGICK o PAINT SHOP PRO. Nuestra experiencia es que en ocasiones, sobre todo con gráficos en color, los EPS creados generan problemas, la mayor parte de las veces porque el lenguaje PostScript que manejan algunas de estas aplicaciones no es totalmente estándar o la BoundingBox no ha sido adecuadamente calculada. Cuando esto ocurra una solución puede ser recurrir a imprimir el archivo gráfico, pero no a papel, sino a un archivo, utilizando, para dicha impresión, un controlador de impresora PostScript lo más estándar posible. Este controlador debe ser de impresora en color si el gráfico posee esta cualidad, y debemos, por supuesto, atender también a la resolución. En nuestra experiencia, las impresoras en color de Apple e IBM suelen dar buen resultado.

Otras herramientas

Existen algunos paquetes y ejecutables que aportan soluciones, casi siempre parciales, al problema de la incompatibilidad de los formatos gráficos. Aquí comentamos muy brevemente algunos de ellos.

PSfrag La utilidad fundamental de este paquete es evitar que el texto de los gráficos PostScript utilice tipos diferentes a los del texto usual del documento. La idea de funcionamiento es sencilla e inteligente. Opera de la forma siguiente. En el momento de generar el gráfico colocaremos un texto ficticio, uno distinto para cada porción de texto que se desee insertar en el gráfico. Cuando se genera la salida PostScript, a través de DVIPS, éste actúa sustituyendo cada uno de los «textos ficticios» por el sustituto, en L^AT_EX, que hayamos indicado. La sus-

titución se hace sobre la marcha, es decir, no se cambia el archivo gráfico. Puede utilizarse con todos los archivos PS, generados por cualquier aplicación, siempre que respeten unas normas mínimas. Para los detalles puede consultar [23].

`epstopdf` Este paquete actúa en el momento de la compilación con PDFLATEX. Cuando al compilar el documento se encuentra con la inclusión de un archivo EPS, la compilación se detiene temporalmente, se convierte el archivo EPS a formato PDF, sobre la marcha, y continúa la compilación. Una segunda compilación incluirá la versión PDF del gráfico. Requiere configurar el compilador adecuadamente. Consulte [49] para más detalles.

`pdftricks` Sigue un proceso similar al del caso anterior, pero referido únicamente a gráficos escritos en código PSTricks. Viene acompañado del script PST2PDF que permite realizar el proceso interno de conversión de forma automática. No contempla todas las posibilidades de PSTricks, por ejemplo, las conexiones de nodos fuera de entornos `pspicture` (véase la página 350). Consulte [12].

`ps4pdf` Este reciente paquete proporciona una nueva opción para utilizar el lenguaje PostScript (gráficos PSTricks, sustituciones PSfrag, gráficos EPS, etc.) con PDFLATEX. Consulte [48].

4.2. El paquete `graphicx`

EN la lección 9 ya nos introdujimos en la incorporación de gráficos externos con la ayuda del paquete `graphicx`. Allí pudimos comprobar que este paquete alcanza un gran nivel de estandarización en dicha tarea, ya que reduce la adaptación al programa visualizador/impresor mediante el cambio de una única opción del propio paquete. El paquete `graphicx` fue creado por D. Carlisle y S. Rahtz [10].

En esta sección completaremos la información acerca de `graphicx`, incluyendo aquellas opciones generales del paquete y del comando `\includegraphics` que no fueron presentadas en la lección 9, así como las herramientas para rotación y escalado de texto, aspectos interesantísimos y sencillos de manejar.

4.2.1. Opciones de `graphicx`

En la sección 3.1, concretamente en el cuadro 3.1, se incluye la lista de las opciones de `graphics` referidas al controlador, que coincide con la de los paquetes `color` e `hyperref`.

A estas opciones de controlador del paquete `graphicx` hay que añadir otras referentes al tratamiento de gráficos, escalado y rotación. Éstas son las siguientes:

`[final|draft]` La opción `draft`, que ya citábamos en la página 60, no imprime los gráficos incluidos con `\includegraphics`. En su lugar dibuja un rectángulo, con las dimensiones del gráfico, en cuyo interior se imprime el nombre del archivo gráfico incluido. Esta opción es aconsejable en las versiones preliminares del documento, ya que la inclusión de todos los archivos gráficos consume mucho tiempo y ralentiza las compilaciones. La opción por defecto es `final`, opuesta a `draft`; esta opción `final` puede ser útil cuando deseamos imprimir los gráficos,

pero también queremos compilar el documento con opción `draft` en la clase de documento, que, de incluirse, se «transmitiría» a `graphicx`.

`hiderotate` Esta opción no muestra el texto rotado; es útil cuando trabajamos con un controlador que no permite esta clase de operaciones.

`hidescale` Similar a la opción anterior, ésta no muestra el texto escalado.

`hiresbb` Cuando se trata de incorporar un gráfico PostScript esta opción le indica al paquete que, en lugar de determinar las dimensiones de todos los gráficos con la línea `%%BoundingBox` del archivo gráfico, como es habitual, lo haga a partir de la línea `%%HiResBoundingBox`.

4.2.2. Todos los parámetros para la inclusión de gráficos externos

Como ya decíamos en el apartado I.9.3, el paquete `graphicx` basa la estrategia de inclusión de gráficos externos en un único comando, a saber comando `\includegraphics`, que se transforma, internamente, en acciones distintas según el controlador. Recordemos brevemente que la sintaxis de este comando es

`\includegraphics[ListaOpciones]{Archivo}`

donde *ListaOpciones* debe atenerse a una sintaxis especial, véase la página 74; concretamente esta sintaxis es de la forma:

Parám1=Valor1 , Parám2=Valor2 , Parám3=Valor3 , . . . , ParámN=ValorN

siendo los parámetros *Parám1*, etc., características, especificadas en el paquete, que permiten modificar la forma en que se incluirá el gráfico que contiene *Archivo*. Recordemos que en el citado apartado de la lección 9 ya comentábamos el significado de los parámetros `width`, `height`, `keepaspectratio`, `scale`, `clip` y `draft`.

A continuación describimos el resto de los parámetros disponibles. Estos parámetros son ejecutados de izquierda a derecha, es decir, en el orden de escritura en el argumento *ListaOpciones*. Esto implica que el orden puede afectar al resultado. Por ejemplo, `[angle=90,height=2cm]` significa que se va a rotar el objeto 90° y, después, se escalará para tener una altura de 2 cm, en cuyo caso, la anchura final del gráfico impreso dependerá de su altura original; sin embargo, el argumento `[height=2cm,angle=90]` escalará primero el gráfico, hasta una altura de 2 cm y posteriormente lo rotará 90° , lo que significa que la anchura total del gráfico impreso será de 2 cm y su altura dependerá de la anchura original.

`totalheight` Define la altura total (`height+depth`) del gráfico y difiere de la altura de la caja si se ha producido alguna rotación. Por ejemplo, si se produce una rotación de -90° en la figura, entonces ésta tendrá altura (`height`) cero, pero su profundidad (`depth`) será grande.

b) El valor que se debe asignar a este parámetro es una lista de cuatro longitudes, separadas por un espacio en blanco. Este parámetro permite imprimir tan sólo una parte del gráfico, algo que no podíamos hacer con los parámetros presentados en la lección 9. Las dos primeras longitudes indican la posición de la esquina inferior izquierda del gráfico que se insertará, medida desde la esquina inferior izquierda de la página. Las dos últimas longitudes indican la posición de la esquina superior derecha, medida respecto al mismo punto. Si en alguna de las cuatro longitudes se omite la unidad de longitud, la unidad seleccionada será el «punto»

PostScript» o bp (72 bp = 1 in). Por ejemplo, si un gráfico EPS posee una BoundingBox igual a 10 20 50 80, el valor bb=30 50 50 80 imprimirá sólo el cuadrante superior derecho de este gráfico.

[natwidth][natheight] Representan conjuntamente una versión simplificada del parámetro bb cuando ésta se utiliza con las dos primeras longitudes nulas; así, bb=0 0 w h es equivalente a natwidth=w,natheight=h.

[viewport] Este parámetro tiene un significado similar al de bb y puede tomar el mismo tipo de valores. Sin embargo, ahora las esquinas de la parte del gráfico que se imprime se miden no respecto a la esquina inferior de la página, sino respecto a la esquina inferior absoluta del gráfico, es decir, en el caso de un EPS, las dos primeras cifras de la línea %%BoundingBox (expresadas éstas en puntos PostScript). En el ejemplo citado en la descripción de bb, un valor viewport=20 30 40 60 seleccionará el mismo cuadrante.

[trim] Este parámetro admite como valores cuatro longitudes (con los mismos requisitos que bb). Cada una de ellas especifica lo que se recortará por cada uno de los lados del gráfico, comenzando en los bordes de gráfico (la BoundingBox). Por ejemplo, trim=10 10 15mm 20mm recorta la figura 10 bp por la izquierda, 10 bp por abajo, 15 mm por la derecha y 20 mm por la parte superior. Si alguna longitud es negativa indica una longitud añadida.

[angle] Indica el ángulo de rotación, que estará expresado en la unidades activas (por defecto, grados sexagesimales).

[origin] Define el punto origen que servirá como referencia en las rotaciones. Puede tomar como valor una combinación de uno o dos caracteres (sin espacio intermedio), cada uno de ellos tomado de uno de los grupos siguientes:

l, r, c con significado claro: izquierda, derecha, centro;

t, b, B que indican, respectivamente, la parte superior de la caja, la parte inferior de la misma y la línea base.

De esta forma se puede seleccionar cada uno de los nueve «puntos cardinales» de la caja que contiene al gráfico, así como tres adicionales sobre la línea base; el punto seleccionado será el centro de la rotación indicada por angle. En general se requiere considerable experiencia en el uso de angle y origin, combinados con los parámetros que especifican dimensiones (width, height, totalheight, etc.).

[hiresbb] Es una variable lógica. Si su valor es true L^AT_EX buscará las dimensiones del gráfico en la línea %%HiResBoundingBox en lugar de hacerlo, como es habitual, en la línea %%BoundingBox.

[type] Especifica la familia a la que pertenece el fichero gráfico que deseamos incorporar. Si se utiliza este parámetro, o cualquiera de los tres siguientes, el argumento *Archivo* del comando \includegraphics no debe contener la extensión del archivo.

[ext] Especifica la extensión del archivo gráfico. Sólo puede usarse junto con type.

[read] Especifica el fichero externo que debe leerse para determinar las dimensiones del gráfico. Sólo puede usarse junto con type.

[command] Especifica cualquier comando que deba ser aplicado al fichero gráfico. Sólo puede usarse junto con type.

EJEMPLO 4.1

```
\newcommand{\TT}{\hspace{1pt}}
\newcommand{\puzzl}[1]{%
\includegraphics[height=1cm,bb=#1,clip]{puzzle.eps}}%
\begin{center}
\puzzl{0 98 133}\TT\puzzl{196 0 293 133}\TT%
\puzzl{196 266 293 399}\TT
\puzzl{98 0 196 133}\TT\puzzl{196 133 293 266}\TT%
\puzzl{98 133 196 266}\TT
\puzzl{98 266 196 399}\TT\puzzl{0 133 98 266}\TT%
\puzzl{0 266 98 399}\%
\par ¿Reconoce al personaje?
\end{center}
```



¿Reconoce al personaje?

Inclusión de gráficos con la opción `bb`, este resultado no podríamos haberlo obtenido con las opciones de la lección 9; el fichero EPS incluido marca las dimensiones `%%BoundingBox: 0 0 293 400`

4.2.3. Rotación y escalado de objetos

Rotar y escalar cualquier objeto, ya sea texto, cajas, gráficos, etc., es enormemente sencillo con el paquete `graphicx`, tal y como se describe a continuación.

Para rotar un objeto disponemos del comando `\rotatebox`:

```
\rotatebox[ListaOpciones]{Ángulo}{Objeto}
```

Este comando comienza por colocar el *Objeto* en una caja, de tipo `\mbox`. El argumento *Ángulo* es el ángulo de giro, en grados sexagesimales y en sentido contrario a las agujas del reloj, que se aplicará al *Objeto* que deseamos rotar. El centro de la rotación será el punto de referencia de la caja en la que se almacena el *Objeto* antes de ser rotado. Por ejemplo, mediante el código `\rotatebox{-30}{esto.}`, obtenemos *estoy*.

El argumento optativo *ListaOpciones* sigue una sintaxis idéntica a la del mismo argumento de `\includegraphics`, siendo los parámetros disponibles los siguientes:

- x Se le debe asignar una longitud. Especifica la coordenada horizontal del punto que será considerado el centro de la rotación. El origen de coordenadas, respecto del cual se mide la coordenada x, es el punto de referencia de la caja en la cual se almacena el *Objeto* antes de ser rotado.
- y Tiene el mismo significado que el anterior, pero referido a la coordenada vertical.
- `origin` Es alternativo a la inclusión de los parámetros x e y. Su significado y sus posibles valores son idénticos a los explicados para el parámetro `origin` de `\includegraphics`.
- `units` Especifica la unidad de medida para los ángulos. Concretamente, `units=Núm` indica que una vuelta completa en sentido contrario a las agujas del reloj equivale a *Núm* unidades. Por ejemplo `units=6.283185` no es más que la forma de medir ángulos en radianes.

En el ejemplo 4.2 se muestra el uso de `\rotatebox`.

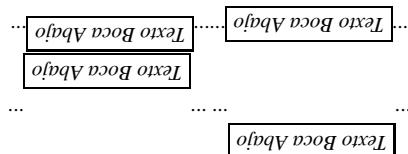
Respecto al escalado de objetos, el paquete proporciona dos métodos diferentes. En el primero se especifica un factor de escala (que puede ser distinto en sentido horizontal y vertical), mientras que el segundo procede cambiando el tamaño de los objetos hasta alcanzar las dimensiones especificadas por el usuario.

EJEMPLO 4.2

```
\def\textof{\fbox{\it Texto Boca Abajo}}
\begin{center}
\ldots\rotatebox{180}{\textof}\ldots
\ldots\rotatebox[origin=c]{180}{\textof}\ldots
\ldots\rotatebox[origin=tr]{180}{\textof}\ldots
\ldots\rotatebox[origin=b,%
  units=6.283185]{3.141592}{\textof}\ldots
\end{center}
```

Parámetros útiles cuando queremos rotar un objeto

Observemos las distintas posiciones, según el valor de los parámetros opcionales, de la frase *Texto Boca Abajo* cuando se gira 180 grados sexagesimales.



El primer método se obtiene mediante el comando

```
\scalebox{EscalaHorizontal} [EscalaVertical] {Objeto}
```

siendo evidente el significado de los argumentos. Si se omite el factor *EscalaVertical*, se toma igual a *EscalaHorizontal*. Cuando se especifican ambos factores, y son distintos, el texto aparece distorsionado, como ocurre en **estas tres palabras**. Este comando también procede insertando previamente el *Objeto* en una caja de tipo `\mbox`.

Un efecto curioso, y probablemente práctico, se obtiene con `\scalebox{-1}[1]{Objeto}`, que imprime el *Objeto*, perdón *Objeto*, reflejado horizontalmente. El comando

```
\reflectbox{Objeto}
```

es exactamente una abreviatura de `\scalebox{-1}[1]{Objeto}`.

El segundo método de escalado se obtiene mediante el uso de las dos sintaxis siguientes:

```
\resizebox{Ancho}{Alto}{Objeto}
```

```
\resizebox*[Ancho]{Alto}{Objeto}
```

Ambos cambian el tamaño del *Objeto*, después de guardarla en una caja, hasta alcanzar una anchura igual a *Ancho* y una altura igual a *Alto*. Si a uno de los argumentos *Ancho* o *Alto* se le asigna como valor el carácter `!`, omitiendo la longitud correspondiente, entonces L^AT_EX calculará dicho argumento para que el resultado final tenga las mismas proporciones que el *Objeto* original, es decir, sin que se produzca distorsión. Normalmente, *Alto* sirve para indicar la altura de la caja (`height`), aunque si se utiliza la versión estrellada, entonces se refiere a la altura total de la caja (`height+depth`). Como es normal para todos los comandos de L^AT_EX que trabajan con cajas, en los argumentos *Ancho* y *Alto*, se pueden utilizar las dimensiones `\height`, `\width`, `\totalheight` y `\depth` de la caja original.

EJEMPLO 4.3

```
\noindent\resizebox{7cm}{\height}{%
  \begin{minipage}{\width}
    \begin{minipage}{\leftmargin}
      \begin{minipage}{\rightmargin}
        Esto es una prueba
      \end{minipage}
    \end{minipage}
  \end{minipage}
}%
\scalebox{-1}{\scalebox{3}{\begin{minipage}{\width}
  \begin{minipage}{\leftmargin}
    \begin{minipage}{\rightmargin}
      Es muy fácil
    \end{minipage}
  \end{minipage}
\end{minipage}}}
\resizebox{5cm}{!}{muy fácil}
```

Cambiando las dimensiones naturales de un objeto

**Esto es una prueba
muy fácil**

4.3. Más herramientas para rotar: el paquete `rotating`

El paquete `rotating`, de Sebastian Rahtz y Leonor Barroca [56], define nuevos entornos y comandos que permiten rotar fácilmente cualquier objeto en L^AT_EX. Nuestro principal interés en este paquete reside en los dos entornos `sidewaysfigure` y `sidewaystable`, que describimos a continuación.

Si utilizamos las herramientas para rotar de `graphicx` (o los entornos `rotate` o `turn` que proporciona el paquete `rotating`) dentro de un entorno flotante (por ejemplo, un entorno `figure` o `table`), el contenido del entorno flotante se verá afectado por la rotación; no así la leyenda, que seguirá imprimiéndose en modo horizontal. Sin embargo, en ciertas ocasiones puede ser conveniente que tanto el objeto flotante como su leyenda sean rotados. Pensemos en un entorno flotante que se debe imprimir girado 90 grados, porque excede de la anchura normal de nuestra página de texto, y que necesita una página completa para él solo. Para este menester, para girar entornos flotantes (incluyendo la leyenda), el paquete dispone de los siguientes entornos:

<pre>\begin{sidewaysfigure} Objeto \end{sidewaysfigure}</pre>	<pre>\begin{sidewaystable} Objeto \end{sidewaystable}</pre>
---	---

La figura I.7.1 de la página 66, por ejemplo, ha sido generada mediante un `sidewaysfigure` y un comando `\caption`.

Para girar las leyendas 90 grados, independientemente de que se rote la figura o no, se debe sustituir, en cualquiera de los entornos flotantes convencionales, el comando `\caption` por el siguiente:

<pre>\rotcaption[<i>TextoLeyendaÍndice</i>]{<i>TextoLeyenda</i>}</pre>
--

Este comando está especialmente indicado cuando el resultado que se obtiene con un entorno del tipo `sidewaysfigure` o `sidewaystable` no es todo lo satisfactorio que deseamos. En estos casos, podemos girar por separado el contenido del entorno flotante y la etiqueta, utilizando `\rotcaption` en lugar del clásico `\caption`.

4.4. Introducción a `PSTricks`

En la primera lección señalábamos los ficheros PostScript como una de las salidas de alta calidad de impresión para nuestras compilaciones con L^AT_EX o como unos ficheros previos para obtener ficheros PDF. Por otra parte, en la lección 9 aprendimos a incorporar en nuestros documentos gráficos PostScript (PS y EPS) proporcionados por aplicaciones externas.

Otra posibilidad para crear e incluir gráficos PostScript en nuestros documentos consiste en incluir las directivas de este lenguaje gráfico en el documento fuente mediante comandos L^AT_EX. Este tipo de comandos son los que proporciona la colección de paquetes de `PSTricks` (listada en el apartado 4.4.4), creada por Timothy van Zandt en 1993 y mantenida, actualmente, por Denis Girou y Sebastian Rahtz.

En líneas generales, los paquetes de `PSTricks` permiten realizar las siguientes tareas:

- dibujar líneas, polígonos, círculos y curvas;
- colocar, escalar y manipular objetos L^AT_EX;
- realizar gráficas de funciones o de listas de datos, incluyendo ejes etiquetados;
- colorear líneas, letras y regiones;
- realizar diagramas de nodos con conexiones, incluyendo estructuras de árboles;
- crear ficheros PostScript EPS con el compilador L^AT_EX.

El compilador T_EX es el programa que produce los gráficos PostScript a la vez que va confecionando nuestro documento. Después, los gráficos aparecerán incluidos en nuestros documentos en el momento de crear los correspondientes ficheros PostScript con el programa DVIPS.

Para trabajar a lo largo de esta sección vamos a utilizar el paquete `pst-all` que carga casi todos los paquetes de PSTRicks.

4.4.1. Nociones básicas

Aunque los comandos de PSTRicks funcionan en cualquier parte del documento, a la hora de diseñar un gráfico es conveniente delimitarlo en una región particular de la página (una caja) que estará reservada para el dibujo. Con PSTRicks tenemos el entorno `pspicture` para realizar esta tarea; su sintaxis es:

```
\begin{pspicture}[Posición] (x0,y0) (x1,y1)
Objeto1
Objeto2
...
\end{pspicture}
```

El entorno construye una caja cuya esquina inferior izquierda tiene coordenadas (x_0, y_0) y la esquina superior derecha viene dada por las coordenadas (x_1, y_1) . Por defecto la unidad a la que se refieren estas coordenadas es 1 cm. También por defecto la caja creada se posiciona sobre la línea base de texto. Esta posición puede modificarse con el argumento optativo *Posición* que es un número cualquiera, no necesariamente entre 0 y 1, aunque el 0 significa que la línea base pasa por la base de la caja, el 1 que pasa por su parte superior, y con los valores comprendidos entre 0 y 1 se puede conseguir que la línea atraviese la caja por cualquier posición.

PSTRicks usa un sistema de variables donde almacena los parámetros de los gráficos. Para modificar los valores asignados por defecto, proporciona el comando

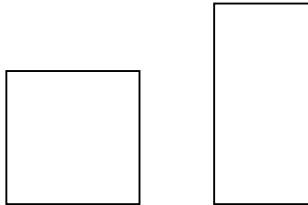
```
\psset{Parám1=valor1,Parám2=valor2...}
```

que tiene por argumento una lista de asignaciones de valores a parámetros separada por comas. Por ejemplo, al escribir `\psset{unit=1mm, linecolor=blue}` establecemos que la longitud de la unidad por defecto va a ser 1 mm y que el color, también por defecto, de las líneas será el azul.

Además del parámetro `unit`, que establece la unidad por defecto para todas las longitudes de PSTRicks, otras unidades básicas que se pueden utilizar son `xunit` e `yunit` para declarar distintas unidades para los ejes coordinados, de manera que se puede reescalar la caja definida por `pspicture` cambiando independientemente la escala de cada eje.

EJEMPLO 4.4

```
\psset{unit=4mm}
\begin{pspicture}(0,0)(5,5)
\psframe(0,0)(5,5)
\end{pspicture}
\psset{xunit=3mm,yunit=6mm}
\begin{pspicture}(0,0)(5,5)
\psframe(0,0)(5,5)
\end{pspicture}
```



PSTricks también puede utilizar coordenadas polares. Basta con realizar la declaración

`\SpecialCoor`

para que la referencia a los puntos se realice en la forma (r, θ) , donde r es la distancia al origen de coordenadas y θ es el ángulo que forma el vector de posición del punto con el eje horizontal.

La unidad de medida de los ángulos, por defecto, son los grados sexagesimales (360 grados para la circunferencia). Ésta puede cambiarse con los comandos siguientes:

`\degrees [Número]`

`\radians`

`\degrees` determina el *Número* de grados de toda la circunferencia, mientras que el segundo expresa los ángulos en radianes (equivale a `\degrees [6.28319]`).

Otro parámetro habitual en los objetos PSTricks es el color. Al utilizar el paquete `pst-all` tenemos a nuestra disposición todos los colores del paquete `color`. En cualquier caso, PSTricks tiene definidos los siguientes colores básicos: `red` (rojo), `green` (verde), `blue` (azul), `cyan` (azul celeste), `magenta` y `yellow` (amarillo); y los colores de la escala de grises: `black` (negro), `darkgray` (gris oscuro), `gray` (gris), `lightgray` (gris claro) y `white` (blanco). Para definir nuevos colores también proporciona el siguiente comando:

`\definecolor{Nombre}{Modelo}{Especificación}`

donde los modelos son los mismos utilizados en el paquete `color` (véase la sección 3.1). Por ejemplo, con la declaración `\definecolor{migris}{gray}{0.4}` definimos un nuevo gris.

4.4.2. Algunos objetos gráficos

Una vez definida la caja `pspicture` donde poner los dibujos, vamos a presentar algunos de los objetos que podemos crear. Todos los comandos que vamos a describir pueden incluir como argumento optativo una lista de asignaciones de valores a los *Parámetros* que determinan el aspecto de los gráficos. Si no se realizan, se utilizarán los valores por defecto. Una alternativa, útil cuando se van a representar varios objetos con las mismas propiedades, es utilizar el comando `\psset` para modificar los valores por defecto.

En los ejemplos de este apartado vamos a hacer algunas asignaciones, aunque dejamos para el siguiente apartado la descripción de los parámetros comunes a todos los objetos. Otra característica común a todos los comandos que determinan curvas es que tienen una versión con asterisco (*) para rellenar el área limitada por la curva usando el color determinado para la curva.

Líneas y poligonales

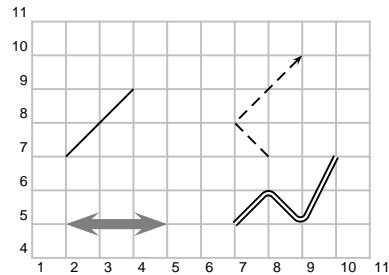
Para dibujar segmentos de línea, vectores y curvas poligonales con o sin ángulos redondeados, disponemos de un único comando:

```
\psline [Parámetros] {Flecha} (x0,y0) (x1,y1) ... (xn,yn)
```

El argumento obligatorio es la lista de coordenadas de los puntos que definen la poligonal. Si sólo se ponen las coordenadas de dos puntos, se representará un segmento o una flecha. La asignación del tipo de *Flecha* a representar, aunque encerrada entre llaves, es optativa; si no se incluye toma el valor por defecto del parámetro *arrows* (véanse el siguiente apartado y el cuadro 4.2).

EJEMPLO 4.5

```
\psset{unit=5mm}
\begin{pspicture}(1,4)(11,11)
\psline(2,7)(4,9)
\psline[linewidth=0.3, linecolor=gray]{<->}(2,5)(5,5)
\psline[linestyle=dashed, arrows=->](8,7)(7,8)(9,10)
\psline[linearc=0.2, doubleline=true](7,5)(8,6)(9,5)(10,7)
\end{pspicture}
```



Ejemplos de líneas poligonales. La cuadrícula se ha añadido para referenciar las coordenadas

Puntos y curvas

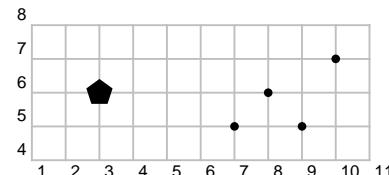
Para dibujar puntos en las coordenadas (x_i, y_i) de una lista, disponemos del comando

```
\psdots [Parámetros] (x0,y0) (x1,y1) (x2,y2) (x3,y3) ...
```

En el argumento *Parámetros* podemos asignar el estilo de los puntos (*dotstyle*) que determina el símbolo a utilizar para dibujarlos, su tamaño (*dotsize*), etc. (véanse el apartado siguiente y el cuadro 4.3 para conocer distintas opciones).

EJEMPLO 4.6

```
\begin{pspicture}(1,4)(11,8)
\psdots[dotstyle=pentagon*, dotsize=10pt](3,6)
\psdots(7,5)(8,6)(9,5)(10,7)
\end{pspicture}
```



Dibujando puntos

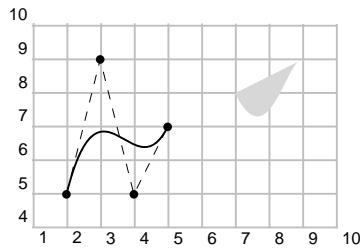
El tipo de curvas bezier, obtenidas mediante interpolación cúbica de cuatro puntos, está disponible en PSTRicks con el comando

```
\psbezier [Parámetros] (x0,y0) (x1,y1) (x2,y2) (x3,y3)
```

EJEMPLO 4.7

```
\psset{unit=5mm}
\begin{pspicture}(1,4)(10,10)
\definecolor{migris}{gray}{0.84}
\psbezier[showpoints=true](2,5)(3,9)(4,5)(5,7)
\psbezier*[linecolor=migris](7,8)(8,6)(8.5,9)(9,9)
\end{pspicture}
```

Ejemplos de curvas bezier



Otro tipo general de curvas, definidas por una lista de puntos, se obtiene mediante interpolación con «splines». En PSTricks se consiguen con los comandos

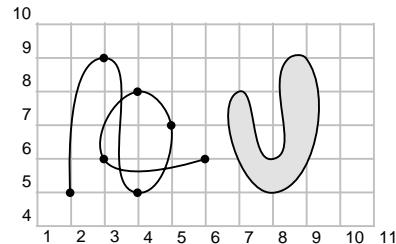
```
\pscurve [Parámetros] (x0,y0) (x1,y1) ... (xn,yn)
\psecurve [Parámetros] (x0,y0) (x1,y1) ... (xn,yn)
\psccurve [Parámetros] (x0,y0) (x1,y1) ... (xn,yn)
```

En la variante \psecurve sólo se traza la curva entre (x_1, y_1) y (x_{n-1}, y_{n-1}) aunque se usan el primer y el último punto para determinar la curva. La otra variante, \psccurve, traza una curva cerrada enlazando el último punto con el primero.

EJEMPLO 4.8

```
\psset{unit=5mm}
\begin{pspicture}(1,4)(11,10)
\pscurve[showpoints=true](2,5)(3,9)(4,5)%
(5,7)(4,8)(3,6)(6,6)
\psccurve[fillstyle=solid,fillcolor=yellow]%
(7,8)(8,6)(8.5,9)(9,9)(8,5)
\end{pspicture}
```

Curvas



Para dibujar estas últimas curvas tenemos que escribir las coordenadas de todos los puntos que las definen. Pero si trabajamos con un programa de cálculo numérico y obtenemos una curva, ésta consistirá en una lista de coordenadas de puntos, previsiblemente de muchos puntos. En este caso no es razonable escribir las coordenadas punto a punto. Para evitarlo, disponemos de los dos comandos siguientes que permiten incorporar datos a un documento L^AT_EX:

```
\savedata{\VariableDatos}[Datos] \readdata{\VariableDatos}{FicheroDatos}
```

Los dos comandos crean el comando \VariableDatos donde se almacenan los datos. Con el primer comando los incluimos «copiando y pegando» desde la salida del programa de cálculo, hasta el argumento *Datos*. El segundo comando va a leer los datos al archivo *FicheroDatos* generado por el programa de cálculo.

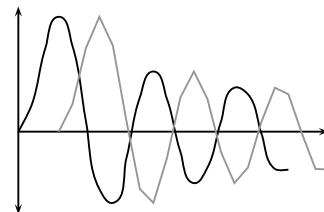
Una vez incorporados los datos, para dibujar la curva disponemos del comando

```
\dataplot [Parámetros]{\VariableDatos}
```

Podemos indicar el estilo de unión entre los puntos de la lista mediante el parámetro `plotstyle` que, por defecto, tiene asignado el valor `line` para unir los puntos mediante segmentos. Otros valores admitidos para este parámetro son: `dots` (líneas de puntos), `curve`, `ecurve` y `ccurve` con el mismo significado que en el comando `\pscurve` y sus dos variantes. También podemos declarar dónde está el origen para referenciar los datos en el parámetro `origin={ x_0, y_0 }` (por defecto, `{0, 0}`).

EJEMPLO 4.9

```
\savedata{\misdatos}[\{{0, 0}, {1, 0.1149},
... , {20, -0.1603}\}]
\readdata{\masdatos}{bessel.dat}
\psset{xunit=.2cm,yunit=3.5cm}
\begin{pspicture}(0,-0.35)(20,.53)
\dataplot[plotstyle=curve]{\misdatos}
\psline[<->](0,-0.35)(0,.53)\psline[->](0,0)(23,0)
\dataplot[origin={-3,0},linecolor=green]{\masdatos}
\end{pspicture}
```



Curva generada en otra aplicación, representada con estilos y orígenes distintos

He aquí un pequeño análisis de este ejemplo. Con el programa MATHEMATICA, o con otro programa de cálculo, generamos una lista de puntos de una de las funciones de Bessel, utilizando llaves para limitar las coordenadas de los puntos y los elementos de la lista, y comas para separar todas las componentes:

```
%% bessel.dat
{{0, 0}, {1, 0.1149}, {2, 0.3528}, {3, 0.4860}, {4, 0.3641}, {5, 0.0465},
{6, -0.2428}, {7, -0.3014}, {8, -0.1129}, {9, 0.1448}, {10, 0.2546},
{11, 0.1390}, {12, -0.0849}, {13, -0.2177}, {14, -0.1520}, {15, 0.0415},
{16, 0.1861}, {17, 0.1583}, {18, -0.0075}, {19, -0.1577}, {20, -0.1603}}
```

Después guardamos la lista en un fichero de texto con el nombre `bessel.dat`. A continuación incorporamos los datos a nuestro documento de dos formas. Por un lado, cortando y pegando los hemos traído a la variable `\misdatos` y, por otro, hemos leído el fichero de datos en la variable `\masdatos`.

Compilando el código del ejemplo 4.9 obtenemos el correspondiente resultado. Obsérvese que hemos representado las curvas con diferentes estilos y diferentes orígenes de referencia (`curve` produce la curva suave y `line` la poligonal).

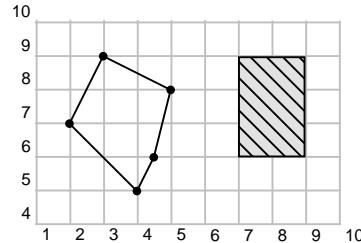
Polígonos, rectángulos, círculos, elipses y arcos

Aunque es posible construir polígonos dibujando una poligonal, el comando `\pspolygon` simplifica la tarea al unir automáticamente el primero y el último de los vértices descritos. Para el caso particular de los rectángulos disponemos del comando `\psframe`, cuyos argumentos son los vértices inferior izquierdo y superior derecho. La sintaxis concreta de estos dos comandos es

```
\pspolygon [Parámetros] ( $x_0, y_0$ ) ( $x_1, y_1$ ) ... ( $x_n, y_n$ )
\psframe [Parámetros] ( $x_0, y_0$ ) ( $x_1, y_1$ )
```

EJEMPLO 4.10

```
\psset{unit=5mm}
\begin{pspicture}(1,4)(10,10)
\pspolygon[showpoints=true]%
(4,5)(2,7)(3,9)(5,8)(4.5,6)
\psframe[fillstyle=vlines*,fillcolor=yellow](7,6)(9,9)
\end{pspicture}
```



Representación de polígonos y rectángulos

Para dibujar elipses tenemos el comando `\psellipse`

```
\psellipse [Parámetros] ( $x_0, y_0$ ) ( $A, B$ )
```

cuyos argumentos son las coordenadas del centro y los semiejes.

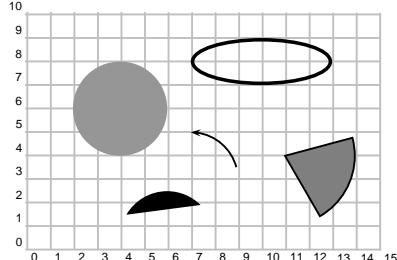
Para trazar círculos, arcos de circunferencia y sectores circulares disponemos de

```
\pscircle [Parámetros] ( $x_0, y_0$ ) {Radio}
\psarc [Parámetros] ( $x_0, y_0$ ) {Radio} {AnguloIni} {AnguloFin}
\pswedge [Parámetros] ( $x_0, y_0$ ) {Radio} {AnguloIni} {AnguloFin}
```

que tienen por argumentos el centro y el radio del círculo, y los ángulos inicial y final que describen el arco (resp. el sector).

EJEMPLO 4.11

```
\psset{unit=3.5mm}
\begin{pspicture}(0,0)(15,10)
\pscircle[fillstyle=solid,fillcolor=green,
linecolor=green](4,6){2}
\psellipse[linewidth=1.5pt](10,8)(3,1)
\psarc[arrows=->](7,3){2}{15}{90}
\psarc*(6,0.5){2}{45}{150}
\pswedge[fillstyle=solid,
fillcolor=gray](11,4){3}{300}{15}
\end{pspicture}
```



Representación de círculos, elipses y arcos

Cuadrículas

En los ejemplos presentados hasta ahora hemos incluido cuadrículas para tener referencia de la posición de los dibujos. Éstas se han obtenido con el comando

```
\psgrid [Parámetros] ( $O_x, O_y$ ) ( $x_0, y_0$ ) ( $x_1, y_1$ )
```

donde el único argumento obligatorio son las coordenadas (x_1, y_1) del vértice superior derecho de la cuadrícula. El argumento (x_0, y_0) representa las coordenadas del vértice inferior izquierdo, si está presente; si no lo está, este vértice se posiciona en el origen $(0,0)$. El argumento (O_x, O_y) indica el punto donde se situarán los ejes etiquetados; por defecto es el vértice inferior izquierdo.

En el ejemplo 4.12 se puede ver el uso de este comando, con alguno de los parámetros específicos que permiten modificar el aspecto de las cuadrículas. Éstos son:

`gridcolor=Color` indica el color de la cuadrícula;

`gridwidth=Longitud` especifica la anchura de las líneas;

`griddots=Número` indica que estas líneas se representan con ese número de puntos en cada intervalo;

`subgriddiv=Número` es el número de subdivisiones de la cuadrícula;

`subgridwidth=Longitud` es el grosor de las líneas de la subdivisión;

`subgridcolor=Color` es el color de las líneas de la subdivisión;

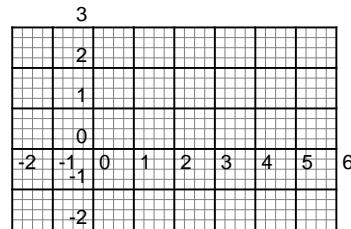
`subgriddots=Número` representa las subdivisiones por líneas de puntos, y el número de puntos por intervalo;

`gridlabelcolor=Color` es el color de los números de los ejes;

`gridlabels=Número` indica la altura de los números de los ejes.

EJEMPLO 4.12

```
\psset{unit=3.5mm}
\begin{pspicture}(-2,-2)(6,3)
\psgrid[gridlabels=0.5,
        subgriddiv=4,
        subgridwidth=0.2pt,
        subgridcolor=gray](0,0)(-2,-2)(6,3)
\end{pspicture}
```



Cuadrículas con PSTRicks

Texto y otros objetos

En cualquier entorno `pspicture` se pueden incluir todo tipo de objetos L^AT_EX. Además podemos adornarlos enmarcándolos dentro de rectángulos, triángulos, rombos, círculos y elipses. A su vez, utilizando los parámetros de estilo, éstos se pueden rotar, rellenar con líneas o colores y sombrear.

Los comandos básicos para posicionar objetos son:

<code>\rput [Posición] {Rotación} (x,y) {Objeto}</code>
<code>\uput [Separación] {Ángulo} {Rotación} (x,y) {Objeto}</code>

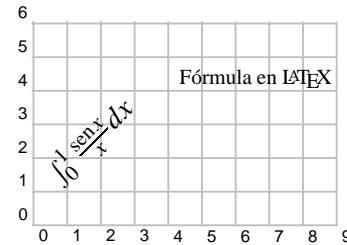
En los dos casos los argumentos obligatorios son el *Objeto* que queremos incluir y las coordenadas (x,y) del punto donde se quiere posicionar. Con `\rput` podemos indicar además la posición del *Objeto* con respecto al punto de coordenadas (x,y) . Esta posición viene dada por una combinación de las iniciales `b,t,l,r` de «bottom» (abajo), «top» (arriba), «left» (izquierda) y «right» (derecha). También podemos hacer una rotación del *Objeto* de ángulo *Rotación*.

El comando `\uput` se utiliza principalmente para poner etiquetas en esquemas y permite indicar la distancia de *Separación* entre el *Objeto* y el punto (x,y) en la dirección dada por el *Ángulo*.

EJEMPLO 4.13

```
\begin{pspicture}(0,0)(9,6)
\rput[b]{45}(1,1){\large
$ \int_0^1 \frac{\sin x}{x} dx $}
\uput{0.5}[-45](4,5){Fórmula en \LaTeX}
\end{pspicture}
```

Objetos en los gráficos PStricks



Los objetos a incluir en el gráfico pueden enmarcarse y adornarse con los siguientes comandos, que también funcionan fuera de `pspicture`:

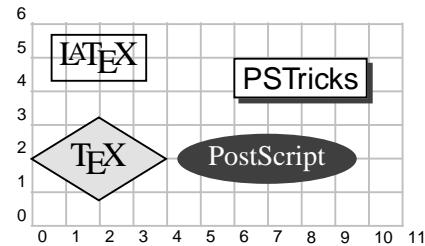
<code>\psframebox [Parámetros] {Objeto}</code> <code>\psdblframebox [Parámetros] {Objeto}</code> <code>\pstribox [Parámetros] {Objeto}</code> <code>\psovalbox [Parámetros] {Objeto}</code>	<code>\pssshadowbox [Parámetros] {Objeto}</code> <code>\psdiabox [Parámetros] {Objeto}</code> <code>\pscirclebox [Parámetros] {Objeto}</code>
--	---

En el ejemplo siguiente mostramos el uso de estos comandos. Obsérvese que se ha usado el comando `\rput` para incluir los objetos, utilizando distintas opciones de posicionamiento.

EJEMPLO 4.14

```
\psset{unit=5mm}
\begin{pspicture}(0,0)(11,6)
\rput(2,5){\psframebox{\Large\LaTeX}}
\rput[tl](6,5){\psframebox[shadow=true]{\Large\PSTricks}}
\rput(2,2){\psdiabox[fillstyle=solid,
fillcolor=yellow]{\Large\TeX}}
\rput(7,2){\psovalbox*[fillcolor=darkgray]{\color{white}\large PostScript}}
\end{pspicture}
```

Más objetos en los gráficos PStricks



Acciones repetidas

Para incluir varias copias de un mismo objeto disponemos de la siguiente variante de `\rput`:

```
\multirput [Posición] {Rotación} ( $x_0, y_0$ ) ( $\Delta x, \Delta y$ ) {NúmVeces} {Objeto}
```

Los argumentos *Rotación* y *Posición* son optativos e indican el ángulo de rotación y la posición del *Objeto*. Tienen la misma sintaxis que en `\rput`. El punto (x_0, y_0) es la referencia para incluir la primera copia del *Objeto*; $(\Delta x, \Delta y)$ indica el incremento en la dirección de cada uno de los ejes, para ir incrementando cada vez las coordenadas del punto de referencia e incluir una copia del *Objeto*; y *NúmVeces* es el número de veces que vamos a repetir el *Objeto*.

Si el objeto a incluir de forma repetida está formado sólo por dibujos PStricks se puede utilizar el siguiente comando:

```
\multips{Rotación}{(x0,y0)}{(Δx,Δy)}{NúmVeces}{Objeto}
```

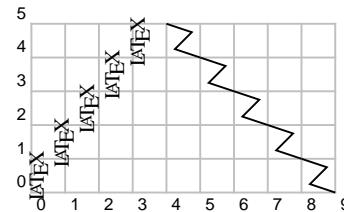
que tiene la misma sintaxis que `\multirput`, con la única diferencia de que este último sólo admite coordenadas cartesianas, mientras que `\multips` también puede utilizar coordenadas polares.

En el ejemplo 4.15 mostramos el uso de estos dos comandos. Obsérvese que el *Objeto* a incluir también puede ser un dibujo PSTRicks, como la curva que aparece en el argumento de `\multips`. En este caso, las coordenadas de las curvas a incluir toman como origen de referencia cada uno de los puntos de inclusión.

EJEMPLO 4.15

```
\begin{pspicture}(0,0)(9,5)
\multirput[b]{90}(0.5,0.5)(0.75,1){5}{\LaTeX}
\multips(4,5)(1,-1){5}%
{\psline(0,0)(0.75,-0.25)(0.25,-0.75)(1,-1)}
\end{pspicture}
```

Repitiendo objetos con PSTRicks



Además de los comandos `\multrput` y `\multips`, PSTRicks proporciona un tercero, implementado por el paquete `multido` (no se carga automáticamente con `pst-all` y debe ser cargado aparte), que permite repetir acciones que dependen de variables, similar al de los lenguajes de programación. Éste es el comando siguiente:

```
\multido{Variables}{NúmVeces}{Acción}
```

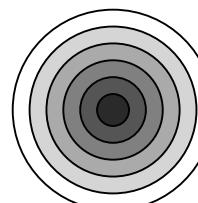
Las variables pueden ser `\d` para una longitud, `\n` para un número (entero o decimal), `\i` para un entero y `\r` para un número decimal (admite sólo cuatro cifras después de la coma). En la declaración se usan expresiones como `\n=ni + δn`, donde la variable `\n` se expresa por su valor inicial, n_i , el signo + y el incremento, $δn$, que hay que efectuar cada vez.

Obsérvese en el ejemplo 4.16, que cuando los incrementos son negativos es necesario escribir dos signos: + para definir el incremento y - para el signo de éste.

EJEMPLO 4.16

```
\psset{unit=1.5cm}
\begin{pspicture}(-0.5,1)(1,1)
\multido{\r=1+-0.1666,\d=0.8+-0.3}{6}{%
\definecolor{migris}{gray}{\r}
\pscircle[fillstyle=solid,
           fillcolor=migris](0,0){\r}
\rput(1.1,\d){r=\r+1}}
\end{pspicture}
```

Repitiendo objetos con multido



r=1.0
r=0.8334
r=0.66681
r=0.50021
r=0.33362
r=0.16702

Creando ficheros EPS

PSTRicks también permite crear gráficos EPS que, tras la correspondiente conversión a PDF, pueden ser utilizados por PDFLATEX o por otras aplicaciones. La forma más simple de producir

buenos ficheros EPS consiste en crear un nuevo fichero `MiEPS.tex` que cargue los paquetes necesarios en el preámbulo y cuyo cuerpo contenga un único entorno `pspicture` con la descripción del gráfico que queramos crear. El entorno `pspicture` estará encerrado por el entorno `TeXtoEPS` siguiendo la siguiente sintaxis:

```
\begin{TeXtoEPS}
  \begin{pspicture}...
  ...
  \end{pspicture}
\end{TeXtoEPS}
```

Después de compilar y obtener el fichero `MiEPS.dvi`, se utiliza el programa DVIPS con la opción `-E` para generar el gráfico `MiEPS.eps`. Esto puede hacerse modificando las opciones de nuestro entorno de trabajo o ejecutando directamente, en una ventana del sistema, la línea de comandos:

`dvips MiEPS -E`

Si el documento fuente sólo contiene un entorno `pspicture` que está bien dimensionado no se suelen producir errores en la creación del gráfico.

Nodos

PSTricks, y de forma más concreta su paquete `pst-node`, ofrece la posibilidad de conectar objetos arbitrarios mediante una curva etiquetada, sin necesidad de conocer cuál es la posición exacta de los objetos ni dónde hay que poner la curva. Las tres componentes de este sistema de «nodos» y «conexiones» son: los nodos (objetos a conectar), las conexiones (curvas entre los nodos) y las posibles etiquetas de las conexiones.

Para declarar que un objeto L^AT_EX es un nodo, basta utilizar el comando

```
\rnode[Posición]{NombreNodo}{Objeto}
```

donde *Posición* tiene la misma sintaxis que en el comando `\rput`. El nombre del nodo, *NombreNodo*, sirve para referirnos a él; debe contener sólo letras y números, comenzando siempre por una letra. Al declarar un *Objeto* como un nodo, L^AT_EX lo escribe o dibuja y guarda las dimensiones de la caja que lo contiene, y la posición del punto de conexión de ésta, determinado por el argumento optativo *Posición*, hacia donde apuntarán las conexiones.

Si se quiere utilizar nodos en un entorno `pspicture` basta con incluirlos dentro del argumento de un comando `\rput`.

Parar realizar las conexiones disponemos de un conjunto de comandos que comparten el inicio de su nombre, `nc` (de «node connection»), y la estructura de su sintaxis:

```
\ComandoConexión [Parámetros] {Flechas} {NodoA} {NodoB}
```

en la que se especifica el tipo de *Flechas* a usar (véase el cuadro 4.2) y los nombres de los nodos a conectar, donde el sentido de la conexión va siempre desde el *NodoA* al *NodoB*.

En el cuadro 4.1 se incluye una brevíssima guía de los comandos de conexión entre nodos que proporciona `pst-node` y de los parámetros que modifican su comportamiento. En los *Parámetros* de todos los comandos de conexión también podemos usar `nodesep=Longitud` para indicar la

anchura del borde añadido alrededor de un nodo, a partir del cual se inician las líneas de conexión (existen versiones independientes para los dos nodos, `nodesepA` y `nodesepB`). Otro parámetro es `offset=Longitud` (`offsetA`, `offsetB`), que produce un desplazamiento del punto de conexión perpendicular a la línea de conexión, en sentidos opuestos según se trate del *NodoA* o del *NodoB*.

Comando	Tipo de conexión entre nodos
<code>\ncline</code>	Línea recta.
<code>\ncarc</code>	Un arco de curva.
<code>\ncdiag</code>	Una línea compuesta de tres piezas rectilíneas. Los brazos que se unen al <i>NodoA</i> y <i>NodoB</i> tienen longitudes respectivas <code>armA</code> y <code>armB</code> , con ángulos de conexión con los nodos <code>angleA</code> y <code>angleB</code> .
<code>\ncdiagg</code>	Como el anterior pero con <code>armB=0</code> .
<code>\ncbar</code>	Tres segmentos rectilíneos, los dos brazos externos perpendiculares al que los une. Se pueden determinar <code>armA</code> y <code>armB</code> (aunque alguno de ellos puede verse aumentado) y, lógicamente, sólo un ángulo (<code>angle</code>).
<code>\ncangle</code>	También un conjunto de tres segmentos. Es posible fijar los ángulos de unión <code>angleA</code> y <code>angleB</code> , así como la longitud <code>armB</code> , pero la longitud <code>armA</code> se determina de modo que el brazo que parte del <i>NodoA</i> forme un ángulo recto con el segmento intermedio.
<code>\ncangles</code>	Similar al anterior, pero los dos brazos se conectan por una línea quebrada compuesta de dos segmentos, formando entre sí y con el brazo en el <i>NodoA</i> sendos ángulos rectos.
<code>\ncloop</code>	Línea quebrada compuesta de cinco segmentos, siendo perpendiculares dos segmentos consecutivos. Si se pone el mismo nodo en los dos extremos modificando los ángulos para que la diferencia <code>angleA-angleB=±180</code> , se obtiene un lazo («loop»).
<code>\nccurve</code>	Curva bezier; los ángulos de incidencia en los nodos se pueden determinar.
<code>\ccircle</code>	Conecta un nodo consigo mismo mediante un arco de circunferencia.
<code>\ncbox</code>	Construye una caja rectangular englobando a los dos nodos.
<code>\ncarcbox</code>	Similar al anterior pero la caja es ovalada.

Cuadro 4.1: Comandos para las conexiones entre nodos

A la hora de poner etiquetas a las conexiones existen muchas posibilidades. Vamos a limitarnos sólo a tres comandos, señalando que hay más posibilidades que pueden consultarse en el manual de PSTRicks [69, 20].

`\naput [Parámetros] {Objeto} \ncput [Parámetros] {Objeto} \nbput [Parámetros] {Objeto}`

`\ncput` coloca la etiqueta encima de la línea, mientras que `\naput` y `\nbput` lo hacen, respectivamente, a izquierda y derecha de ella, considerando que la línea se recorre desde el nodo de origen al de destino.

Estos comandos que colocan una etiqueta a una conexión deben escribirse inmediatamente después del comando que define la conexión.

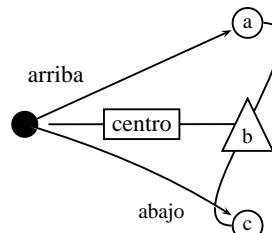
Las etiquetas se sitúan en una posición que, podemos decir, se calcula como una cierta fracción de la longitud total de la conexión. En realidad la situación es más compleja: la posición depende del número de segmentos que posea la conexión (que típicamente varía entre 1 y 5) y del parámetro `npos=Número`, cuyo significado es: si la conexión posee, por ejemplo, cuatro segmentos, entonces `npos` puede variar entre 0 y 4; un valor entre 0 y 1 indica la fracción de longitud del primer segmento, entre 1 y 2, la fracción del segundo segmento, y así sucesivamente. Las etiquetas también pueden rotarse con el parámetro `nrot=Ángulo`.

EJEMPLO 4.17

```

\psset{unit=1cm,fillstyle=solid,fillcolor=white}
\begin{pspicture}(-2,-2)(2,2)
\rput(-1.8,0){\rnode{A}{\pscircle*(0,0){0.2}}}
\rput(1.5,1.5){\rnode{a}{\pscirclebox{a}}}
\ncline{->}{A}{a}\nput[npos=0.25]{\small arriba}
\rput(1.5,-1.5){\rnode{c}{\pscirclebox{c}}}
\nccurve[angleA=180,angleB=0]{c}{a}
\rput(1.5,0){\rnode{b}{\pstribox{b}}}
\ncline[nodesepA=10pt,nodesepB=-6pt]{-}{A}{b}
\ncput{\psframebox{\small centro}}
\ncarc{->}{A}{c}\nput[npos=0.75]{abajo}
\end{pspicture}

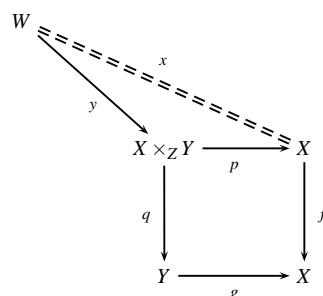
```



Uniendo nodos con PSTricks

Con *pst-node* se pueden realizar diagramas en modo matemático, siendo una mejor herramienta que el paquete *amscd* descrito en la sección 5.11, que sólo permite realizar diagramas conmutativos sencillos.

EJEMPLO 4.18



Un diagramma commutativo con PSTricks

PSTricks proporciona el entorno `psmatrix`, que tiene la misma estructura que `matrix` (véase la sección 5.6) donde cada celda se define automáticamente como un nodo etiquetado por las coordenadas de su posición en la matriz. Además, dentro de este entorno, se pueden poner etiquetas a las conexiones entre nodos siguiendo la sintaxis del modo matemático: `^` o `_` para ponerlas sobre la

conexión o bajo la misma; y $>$ o $<$ para ponerlas a la derecha o a la izquierda de la conexión. Para utilizarlos hay que desactivar algunos métodos taquigráficos de babel (véase la sección 1.2.3); en concreto basta utilizar `\spanishdeactivate{> <}` como en el ejemplo 4.18.

Los diagramas utilizados a lo largo del libro han sido realizados con enlaces PStricks. Véase, por ejemplo, la figura 1.2. Otro ejemplo llamativo del uso de conexiones entre nodos es la presentación 7.10 del capítulo 7, donde se conectan distintas partes de una animación.

4.4.3. Parámetros de PStricks

En este apartado vamos a describir algunos de los parámetros que determinan el aspecto de los gráficos construidos con PStricks. Los valores que tienen asignados por defecto aparecen al final de la descripción entre paréntesis.

`[linewidth]` Es una longitud que indica el grosor de las curvas y líneas (0.8pt).

`[linecolor]` Es el color de las curvas y líneas (black).

`[linestyle]` Es el estilo de las líneas. Sus posibles valores son: `solid`, dibuja la línea continua; `none`, no dibuja la línea; `dashed`, dibuja la curva con segmentos de longitud `Long1` situados a distancia `Long2`, donde los valores de estas longitudes están guardadas en el parámetro `dash=Long1 Long2`; `dotted`, dibuja una línea de puntos situados a distancia `dotsep=Long` (`solid`).

`[linearc]` Representa el valor del radio de la circunferencia utilizada para redondear los ángulos de las poligonales (0).

`[arrows]` Describe los extremos de las curvas o líneas abiertas que le dan aspecto de flecha. Los posibles estilos aparecen en el cuadro 4.2 (-).

`[doubleline]` Si toma el valor «true» se dibujan líneas dobles. La separación entre las líneas viene dada por el parámetro `doublesep` y el color intermedio está determinado por el parámetro `doublecolor` (false).

`[showpoints]` Si toma el valor «true» dibuja todos los puntos utilizados en la definición de la curva (false).

`[dotstyle]` Determina el estilo que se usará al dibujar cualquier punto. Los posibles estilos se indican en el cuadro 4.3. El tamaño de estos puntos puede modificarse en `dotsize`. Con `dotangle` pueden rotarse de forma que, por ejemplo, un cuadrado (`square`) girado 45 grados se convierte en un rombo (*).

Estilo	Resultado	Estilo	Resultado	Estilo	Resultado
-	—	->	→	<->	↔
>-<	→←	<>->	↔	[-]	[]
* - *	—	<->	→	(-)	(→)
-	—	<*-> *	← →	c-c	—
-	•—•	o-o	○—○	cc-cc	—
-	•—•	oo-oo	○—○	C-C	—

Cuadro 4.2: Tipos de flechas en PStricks

Estilo	Punto	Estilo	Punto	Estilo	Punto	Estilo	Punto
*	●			o	○	+	+
x	×	asterisk	*	diamond		diamond*	◆
square		square*	■	oplus	⊕	otimes	⊗
pentagon	◇	pentagon*	◆	triangle		triangle*	▲

Cuadro 4.3: Puntos con PSTricks

[fillcolor] Es el color de relleno para las regiones de PSTricks (white).

[fillstyle] Determina el estilo de relleno para las regiones cerradas. Los posibles estilos son: none, solid, vlines, vlines*, hlines, hlines*, crosshatch, crosshatch* y gradient. El estilo none no hace nada. vlines, hlines, crosshatch utilizan, respectivamente, líneas verticales, horizontales o cruzadas, para llenar las regiones. Con vlines*, hlines*, crosshatch* se usa el estilo conjuntamente con el parámetro fillcolor. Las líneas utilizadas en estos estilos se pueden rotar, separar y colorear con los valores de los parámetros hatchangle, hatchsep y hatchcolor. De hecho, el valor predeterminado de hatchangle es 45, por lo que vlines y hlines no son originalmente verticales ni horizontales. El estilo gradient está disponible en el paquete pst-grad, y rellena las regiones con colores que van cambiando gradualmente de un color a otro; al utilizarlo también debemos asignar los colores inicial gradbegin y final gradend, como en el ejemplo 4.19 (none).

EJEMPLO 4.19

```
\begin{pspicture}(0,0)(25.00,60.00)
\rput(25.00,60.00){\psovalbox[linewidth=0.15,
    linecolor=white, fillstyle=gradient,
    gradbegin=yellow,gradend=red]
    {\scalebox{4}{\color{white}\bfseries\LaTeX}}}
\end{pspicture}
```

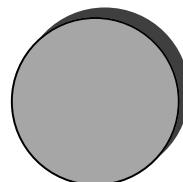


Colores graduados con pst-grad

[shadow] Cuando se le asigna el valor true la región aparece sombreada hasta una distancia del borde dada por shadowsize, en la dirección del ángulo shadowangle y utilizando el color asignado en shadowcolor, como en el ejemplo 4.20 (false).

EJEMPLO 4.20

```
\begin{pspicture}(0,0)(25.00,25.00)
\definecolor{userFillColour}{rgb}{1.00,0.60,0.00}
\pscircle[fillcolor=userFillColour,fillstyle=solid,
    shadow=true,shadowsize=2,shadowangle=45.00,
    shadowcolor=darkgray](12.5,12.5){12.5}
\end{pspicture}
```



Región sombreada con PSTricks

4.4.4. Paquetes de **PSTricks**

Los paquetes que forman **PSTricks** son los siguientes:

- pst-all:** Carga casi todos los paquetes de **PSTricks** y el paquete **pstcol**.
- pstricks:** Contiene la mayor parte de los comandos de **PSTricks**.
- pstcol:** Se distribuye con **graphicx** [10]. Hace **pstricks** compatible con **graphicx** y **color**.
- pst-fr3d:** Para construir cajas tridimensionales.
- pst-3d:** Para hacer gráficos 3-D.
- pst-gr3d:** Para construir cuadrículas 3-D.
- pst-char:** Para colorear y llenar caracteres de texto.
- pst-circ:** Para crear circuitos eléctricos.
- pst-coil:** Para dibujar líneas y colocar objetos en espirales o en zigzag.
- pst-eps:** Para exportar a ficheros PostScript EPS objetos **PSTricks**.
- pst-fill:** Para llenar y colorear regiones arbitrarias.
- pst-grad:** Para graduar los colores de relleno.
- pst-lens:** Para dibujar lentes.
- pst-node:** Para definir nodos y conexiones entre ellos.
- pst-osci:** Para dibujar osciloscopios.
- pst-plot:** Para hacer gráficas 2-D con listas de datos o gráficas de funciones.
- pst-3dplot:** Para hacer gráficas con listas de datos 3-D o gráficas de funciones 3-D.
- pst-poly:** Para dibujar polígonos.
- pst-text:** Para escribir textos a lo largo de curvas.
- pst-tree:** Para realizar estructuras en árbol.
- pst-vue3D:** Para visualizar objetos 3-D.

PARA SABER MÁS

-  En el archivo **MasGraphicx.pdf** puede consultar algunas posibilidades adicionales que aporta el paquete **graphicx**.
-  El paquete **rotating** posee algunos comandos para rotar texto, de índole general, no exclusivos de las leyendas de figuras y cuadros. El archivo **Rotating.pdf** describe estos comandos y contiene algunos ejemplos interesantes.
-  **LATeX** posee herramientas para el diseño de gráficos totalmente estándar, es decir, no dependientes del compilador o del visualizador/impresor. La desventaja es que son muy elementales, por lo que realizar gráficos complicados es lento y difícil. Encontrará toda la información sobre estas herramientas, centradas alrededor del entorno **picture**, en el archivo **Picture.pdf**.

-  El entorno picture posee mejoras, también independientes de la salida, aportadas por el paquete **epic**; consulte el archivo **Epic.pdf**.
-  El paquete **epic** ha sido, a su vez, mejorado, pero ahora de forma no estándar. El paquete **eepic**, descrito en el archivo **Eepic.pdf**, aporta estas mejoras que son comprendidas por DVIPS y DVIPDFM.
- La familia de paquetes de **PSTricks** continúa creciendo y aportando potentes comandos para el diseño de gráficos. En el sitio oficial de la asociación «**TEX Users Group**» se puede seguir su desarrollo, acceder a los manuales de uso y a una fantástica colección de ejemplos (www.tug.org/applications/PSTricks).
- Existen herramientas para el diseño gráfico que guardan los dibujos en formato **PSTricks**, como **JPICEDT**, disponible para ejecutarse sobre el motor de **JAVA** en cualquier plataforma. Para descargarlo y tener información de las últimas modificaciones visite el sitio de Internet de este proyecto hospedado por la organización SourceForge(www.jpicedt.org).

Capítulo 5

Matemáticas

EN este capítulo se analizan con detalle las herramientas que L^AT_EX proporciona para escribir textos de matemáticas con una presentación de alta calidad. Si usted tiene fórmulas matemáticas que escribir, L^AT_EX es la herramienta idónea para presentarlas. Lo único que L^AT_EX no hará por usted es mejorar la calidad científica de sus escritos; esto sigue siendo su responsabilidad.

En la lección 16 aprendimos a declarar el modo matemático, a escribir algunas fórmulas matemáticas sencillas que utilizan subíndices y superíndices, letras griegas, etc. Allí se recomendó la utilización del paquete amsmath para escribir matemáticas, porque en él se incorporan algunos comandos y funcionalidades que facilitan la escritura de fórmulas. Si ahora usted está leyendo este capítulo es porque probablemente necesita:

- Más símbolos y fuentes para sus fórmulas. Por ejemplo, $\mathbb{R}, \mathcal{T}, \partial, \uparrow\uparrow, \prod, \approx, \otimes_{n=1}^{\infty}$, etc.
- Escribir fórmulas más complejas que las estudiadas en la lección 16. Por ejemplo, matrices
$$A - \lambda I = \begin{pmatrix} x - \lambda & 1 \\ 0 & x - \lambda \end{pmatrix},$$
fórmulas centradas en varias líneas, etc.
- Utilizar entornos numerados del tipo teorema:

Teorema 5.1 (Bolzano-Weierstrass). *Los conjuntos cerrados y acotados de la recta son compactos.*

Corolario 5.2. *Las funciones continuas definidas en un intervalo cerrado y acotado de la recta son acotadas.*

Para llevar a cabo todas estas tareas apostamos sin duda por utilizar conjuntamente los paquetes distribuidos por la *American Mathematical Society*: amsmath versión 2.0 (que contiene los paquetes amsbsy, amstext y amsopn) y amssymb (que es un superconjunto del paquete amsfonts que permite utilizar letras «blackboard bold», «fraktur» y todos los símbolos de las fuentes msam y msbm). En todo lo que sigue no distinguiremos entre los comandos intrínsecos de L^AT_EX y las extensiones proporcionadas por amsmath y amssymb. A partir de este momento suponemos siempre que nuestro documento cuenta en su preámbulo con la línea `\usepackage{amsmath, amssymb}`, aun sabiendo que algunas tareas que describiremos a continuación se pueden realizar sin cargar alguno de estos paquetes. La razón para operar así es doble: primero, sencillez para el lector; segundo, porque los paquetes amsmath y amssymb proporcionan mejoras sobre funcionalidades que ofrece

\LaTeX estándar (por ejemplo, un modo fácil de declarar nuevos nombres de operadores similares a arc tg , arc sen , etc., una forma sencilla de subordinar la numeración de las ecuaciones, múltiples sustitutos de los entornos `eqnarray` para fórmulas en varias líneas con un tratamiento excelente de los espacios verticales y horizontales en las mismas, la posibilidad de escribir subíndices en varias líneas, etc.).

5.1. Recordando conceptos básicos

COMO se dijo en la lección 16, para escribir una fórmula, \LaTeX trabaja en modo matemático, que puede ser ordinario o resaltado (fórmulas centradas). El modo matemático ordinario puede ser invocado de las formas que se recuerdan en el cuadro siguiente:

Modo matemático ordinario	Modo matemático resaltado
<code>\$...\$</code>	<code>\[...\]</code>
<code>\(...\)</code>	<code>\begin{displaymath}...\end{displaymath}</code>
<code>\begin{math}...\end{math}</code>	<code>\begin{equation*}...\end{equation*}</code> <code>\begin{equation}...\end{equation}</code>

En modo matemático \LaTeX se ocupa del tamaño de las fuentes, del tamaño de las rayas de fracción o símbolos para raíces, ignora espacios innecesarios, inserta espacios alrededor de operadores, etc. No es recomendable la utilización de `$$...$$` (propia de \TeX , y que algunos usuarios utilizan también con \LaTeX) para invocar el modo matemático resaltado en \LaTeX ; si se utiliza no se recibirá ningún mensaje de error y funcionará correctamente en muchos casos, pero en algunos otros (por ejemplo, si la opción `fleqn` está activa) la distribución de espacios en blanco alrededor de las fórmulas no es satisfactoria. Los tres primeros modos que indicamos para invocar el modo matemático resaltado son equivalentes. La versión sin estrella `\begin{equation}...\end{equation}` numera la fórmula tal y como se explicó en el apartado I.16.1. Esta última versión es la adecuada para el uso de referencias cruzadas con `\label`, `\ref` y `\eqref`.

5.2. Opciones del paquete `amsmath`

EL paquete `amsmath` proporciona algunas características con las que se puede modificar la ubicación por defecto de subíndices y superíndices para operadores, la ubicación de etiquetas para fórmulas numeradas, etc. Las siguientes opciones de `amsmath` activan|desactivan estas características. Como en otras ocasiones, en cada ítem la primera que aparece es la opción por defecto. `centertags|tbtags` Con la primera opción, al escribir una ecuación en más de una línea con el entorno `split` dentro de un entorno `equation`, se coloca el número de la ecuación centrado verticalmente con respecto a las líneas de `split`. El entorno `split` se describe en la sección 5.7.3. Con la segunda, en la situación de antes, se coloca el número de la ecuación en el margen de la última (resp. la primera) línea si estos números van a la derecha (resp. a

	Ord	Op	Bin	Rel	Ape	Cie	Pun
Ord	0	1	(2)	(3)	0	0	0
Op	1	1	*	(3)	0	0	0
Bin	(2)	(2)	*	*	(2)	*	*
Rel	(3)	(3)	*	0	(3)	0	0
Ape	0	0	*	0	0	0	0
Cie	0	1	(2)	(3)	0	0	0
Pun	(1)	(1)	*	(1)	(1)	(1)	(1)

Cuadro 5.1: Separación entre operadores matemáticos

la izquierda). En realidad este resultado se obtiene posicionando la caja definida por `split` sobre (resp. bajo) la línea base de `equation`.

`sumlimits|nosumlimits` Con la primera opción se colocan los límites de todos los operadores de tamaño variable, excluyendo el símbolo integral, centrados bajo o sobre el correspondiente símbolo, cuando se escriben fórmulas centradas. La segunda siempre coloca los límites de los operadores de tamaño variable a la derecha de los símbolos con independencia del tipo de fórmula.

`nointlimits|intlimits` Fijan la posición de los límites del símbolo integral. La primera opción funciona como `nosumlimits` y la segunda lo hace como `sumlimits`.

`namelimits|nonamelimits` La primera opción funciona como `sumlimits` para algunos nombres de funciones como `lím` (`\lim`), `inf` (`\inf`), `máx` (`\max`), etc., que admiten subíndices que aparecen bajo el nombre de la función cuando la fórmula está centrada. La segunda opción coloca los subíndices a la derecha de los nombres de función.

El paquete `amsmath` también admite las siguientes opciones de las clases de documento, que se seleccionan con el comando `\documentclass`.

`leqno|reqno` Coloca los números de las ecuaciones a la izquierda o derecha, respectivamente.

`fleqn` Coloca las fórmulas resaltadas (definidas entre `\[` y `\]`, el entorno `displaymath` o por la familia de entornos relacionados con `equation` descritos en la sección 5.7) a una distancia fija del margen izquierdo, en lugar de centrarlas en la página.

5.3. Símbolos matemáticos

EN los textos de matemáticas, además de fracciones, raíces, exponentes o letras griegas, cuya utilización ya ha sido descrita en la primera parte de este libro, suelen aparecer flechas, símbolos, funciones, etc., que `LATEX` conoce y es capaz de imprimir adecuadamente. El nombre con el que `LATEX` reconoce cada uno de estos objetos suele ser muy próximo al nombre real, en inglés, con el que los designan los matemáticos.

Los símbolos en matemáticas se agrupan en distintas clases que corresponden, más o menos, a la clasificación que éstos tendrían si la fórmula de la que forman parte fuera expresada verbalmente. Son relevantes para esta clasificación los espacios en blanco que se insertan alrededor de los mismos para facilitar la lectura de las fórmulas. Atendiendo a lo anterior, los símbolos se clasifican en las siete categorías que siguen.

Ordinarios (Ord): Por ejemplo, las letras latinas (que, en modo matemático, se escriben en una fuente itálica) A, B, C , etc., a, b, c , etc., letras griegas α, β, γ , etc., números (que en modo matemático utilizan una fuente de tipos rectos) $0, 1, 2$, etc., y los símbolos $\exists, \emptyset, \infty$, etc. Los cuadros I.16.2, 5.3, 5.2 y 5.4 contienen los símbolos ordinarios para L^AT_EX.

Operadores de tamaño variable (Op): Por ejemplo, \sum, \prod, \int , etc. Véase el cuadro 5.18.

Operadores binarios (Bin): Por ejemplo, $+, \cup, \setminus$, etc. El cuadro 5.5 contiene los símbolos de los operadores binarios.

Operadores de relación (Rel): Por ejemplo, los símbolos $=, <, \approx$, etc., las flechas $\circlearrowleft, \longrightarrow, \iff$, etc., los símbolos de inclusión $\subset, \supset, \not\subset$, etc. Los cuadros 5.6, 5.7 y 5.9 contienen los símbolos de los operadores de relación.

Delimitadores de apertura (Ape): Por ejemplo, $(, \{$, etc. Véase el cuadro 5.11.

Delimitadores de cierre (Cie): Por ejemplo, $), \}$, etc. Véase el cuadro 5.11.

Signos de puntuación (Pun): Por ejemplo, $?, ,$, etc. Véase el cuadro 5.14.

El cuadro 5.1 explica cuáles son los espacios que se insertan en una fórmula entre símbolos adyacentes que se han listado de acuerdo con la clasificación anterior. Los números 0, 1, 2 y 3, en el cuadro, se utilizan para indicar, respectivamente, que no se deja espacio, que se deja un espacio pequeño (`\thinspace`, `\,`), un espacio medio (`\medspace`, `\:`) y un espacio grande (`\thickspace`, `\;`). Los espacios correspondientes a las entradas que aparecen en la tabla entre paréntesis se insertan sólo cuando se está en modo `textstyle` o `displaystyle` y no se insertan cuando se está en modo `scriptstyle` o `scriptscriptstyle` (véase la sección 5.5.4). Por ejemplo, muchas de las entradas en la columna y fila Bin son «(2)»: esto significa que en muchos casos se insertará un espacio `\medspace` antes y después de un símbolo como $+$, pero no cuando éste aparezca en subíndices o superíndices. Las entradas que llevan un asterisco * son casos que, simplemente, no se presentan.

La relación de símbolos que mostramos en esta sección no es exhaustiva, aunque cubre todos los símbolos matemáticos disponibles en L^AT_EX básico y en el paquete `amssymb`. Si usted necesita algún símbolo no relacionado aquí, es aconsejable que consulte *The Comprehensive L^AT_EX Symbols list* (Pakin) en CTAN en:

<http://www.ctan.org/tex-archive/info/symbols/comprehensive>

5.3.1. Negación de símbolos

En los cuadros de esta sección aparecen los símbolos de L^AT_EX y las correspondientes negaciones para algunos de ellos. En general, la negación de algunos símbolos se puede obtener también poniendo una barra inclinada sobre los mismos. Esto se consigue con el comando

`\not`

\aleph	<code>\aleph</code>	ℓ	<code>\ell</code>	∂	<code>\partial</code>	\eth	<code>\eth</code>	\wp	<code>\wp</code>	\circledS	<code>\circledS</code>	\Game	<code>\Game</code>
\beth	<code>\beth</code>	\hbar	<code>\hbar</code>	\Bbbk	<code>\Bbbk</code>	\hslash	<code>\hslash</code>	\Re	<code>\Re</code>	\Im	<code>\Im</code>	\mathbb{R}	<code>\mathbb{R}</code>
\daleth	<code>\daleth</code>	\hslash	<code>\hslash</code>	\mathbb{R}	<code>\mathbb{R}</code>	\mathbb{C}	<code>\mathbb{C}</code>	\mathbb{H}	<code>\mathbb{H}</code>	\mathbb{N}	<code>\mathbb{N}</code>	\mathbb{Q}	<code>\mathbb{Q}</code>
\gimel	<code>\gimel</code>	\mathbb{H}	<code>\mathbb{H}</code>	\mathbb{N}	<code>\mathbb{N}</code>	\mathbb{Q}	<code>\mathbb{Q}</code>	\mathbb{R}	<code>\mathbb{R}</code>	\mathbb{Z}	<code>\mathbb{Z}</code>	\mathbb{C}	<code>\mathbb{C}</code>
\complement	<code>\complement</code>	\mathbb{O}	<code>\mathbb{O}</code>	\mathbb{P}	<code>\mathbb{P}</code>	\mathbb{Q}	<code>\mathbb{Q}</code>	\mathbb{R}	<code>\mathbb{R}</code>	\mathbb{N}	<code>\mathbb{N}</code>	\mathbb{Q}	<code>\mathbb{Q}</code>

Cuadro 5.2: Símbolos ordinarios

Γ	<code>\varGamma</code>	Σ	<code>\varSigma</code>
Δ	<code>\varDelta</code>	Υ	<code>\varUpsilon</code>
Θ	<code>\varTheta</code>	Φ	<code>\varPhi</code>
Λ	<code>\varLambda</code>	Ψ	<code>\varPsi</code>
Ξ	<code>\varXi</code>	Ω	<code>\varOmega</code>
Π	<code>\varPi</code>		

Cuadro 5.3: Símbolos ordinarios: fuente itálica de letras griegas mayúsculas

situado justo antes del comando correspondiente al símbolo. Así, por ejemplo, `\not\in` produce el símbolo \notin , o `\not\subset` produce $\not\subset$. L^AT_EX escribe la barra inclinada producida por este comando en una caja de anchura nula, estando su tamaño en correspondencia con el tamaño de los símbolos afectados. Si al utilizar `\not` no nos gusta la posición en la que la barra «/» aparece, ésta puede modificarse con alguno de los comandos de espaciado del cuadro I.16.1, situado entre el comando `\not` y el del símbolo que le sigue.

5.3.2. Redefiniendo símbolos

Se puede hacer que L^AT_EX, en modo matemático, asigne a cualquiera de los símbolos o letras existentes, o a cualquier otro que podamos incorporar, el comportamiento, y por ende los espacios en blanco adyacentes, correspondiente a un símbolo de relación, operador binario, etc. Para ello disponemos de los comandos:

```
\mathord{Expresión}
\mathrel{Expresión}
\mathbin{Expresión}
```

El comando `\mathord` declara *Expresión* como símbolo ordinario, el comando `\mathrel` la declara como un símbolo de relación y por último, `\mathbin` la declara como un símbolo de operación binaria. En la sección PARA SABER MÁS de este capítulo encontrará referencias sobre los comandos `\DeclareSymbolFont` y `\DeclareMathSymbol` con los que se pueden definir todo tipo de símbolos matemáticos. En los pies de los cuadros 5.5 y 5.18 encontrará sendos ejemplos sobre la utilización de estos comandos, en los que se definen, respectivamente, un operador binario y un operador de tamaño variable a partir de un carácter de una fuente.

#	\#	/	\diagup	\neg
&	\&	\diamondsuit	\diamondsuit	\exists
\angle	\angle	\emptyset	\exists	\prime
\backprime	\backprime	\exists	\flat	\sharp
\bigstar	\bigstar	\forall	\forall	\spadesuit
\blacklozenge	\blacklozenge	\heartsuit	\forall	\sphericalangle
\blacksquare	\blacksquare	\infty	\lozenge	\square
\blacktriangle	\blacktriangle	\infty	\measuredangle	\surd
\blacktriangledown	\blacktriangledown	\nabla	\nabla	\top
\bot	\bot	\natural	\natural	\triangle
\clubsuit	\clubsuit			\triangledown
\diagdown	\diagdown			\varnothing

Cuadro 5.4: Símbolos ordinarios: miscelánea. Los símbolos \square y $\#$ se utilizan algunas veces de forma inadecuada como operadores de relación. La utilización de \square y $\#$ como operadores de relación produce un espacio incorrecto, dado que estos comandos proporcionan símbolos ordinarios. Sinónimo: \neg

EJEMPLO 5.1

Si declaramos el signo `+$` como ordinario obtenemos `\((a\mathord{+}b)\)` frente a `\((a+b)\)`. Denotaremos `\(a\mathrel{R}b\)` cuando `a` esté relacionado con `b`. Denotaremos `$A \times B$` es el conjunto de los pares `\((a\mathbin{,}b)\) \dots`

Si declaramos el signo `+` como ordinario obtenemos $(a+b)$ frente a $(a+b)$.

Denotaremos $a R b$ cuando a esté relacionado con b . $A \times B$ es el conjunto de los pares $(a, b) \dots$

Redefinición de símbolos: ordinarios, binarios y de relación

5.4. Fuentes en modo matemático

Como hemos dicho con anterioridad, las letras que son parte de una fórmula aparecen en un tipo de letra itálica. Además de este tipo de letra podemos utilizar, en el modo matemático, los que nos proporcionan los siguientes comandos, que corresponden, en el orden en el que aparecen, a las fuentes: «blackboard bold», negrita, caligráfica, «fraktur», itálica, normal, roman, «sanserif» (sin adornos) y tipo máquina de escribir.

\mathbb{SubFórmula}	\mathnormal{SubFórmula}
\mathbf{SubFórmula}	\mathbf{SubFórmula}
\mathcal{SubFórmula}	\mathcal{SubFórmula}
\mathfrak{SubFórmula}	\mathfrak{SubFórmula}
\mathit{SubFórmula}	\mathit{SubFórmula}

Todos estos comandos admiten como argumento el trozo de la fórmula `SubFórmula` que contiene las letras que deben aparecer en la correspondiente fuente. Los únicos símbolos que responden co-

*	*	◎ \circledcirc	⊖ \ominus
+	+	⊖ \circledddash	⊕ \oplus
-	-	∪ \cup	⊘ \oslash
II ^a	\amalg	⊟ \Cup	⊗ \otimes
*	\ast	⋎ \curlyvee	± \pm
⊓	\barwedge	⋎ \curlywedge	⊸ \rightthreetimes
○	\bigcirc	† \dagger	⊸ \rtimes
▽	\bigtriangledown	‡ \ddagger	⊙ \setminus
△	\bigtriangleup	◇ \diamond	⊜ \smallsetminus
□	\boxdot	÷ \div	⊠ \sqcap
□	\boxminus	* \divideontimes	⊡ \sqcup
□	\boxplus	+ \dotplus	★ \star
☒	\boxtimes	⊠ \doublebarwedge	× \times
•	\bullet	> \gtrdot	⊲ \triangleleft
∩	\cap	T \intercal	▷ \triangleright
⊩	\Cap	⊸ \leftthreetimes	⊕ \uplus
·	\cdot	⊴ \lessdot	⊤ \vee
▪	\centerdot	⊶ \ltimes	⊤ \veebar
○	\circ	⊷ \mp	⊸ \wedge
⊛	\circledast	⊙ \odot	⊸ \wr

^aCuando se tiene cargado el paquete `mathptmx` (versión 2003/03/02 PSNFSS-v9.0c), `\amalg` no está disponible. La salida II corresponde al carácter '161 de la fuente `cmsy` con la codificación «OMS» (véase la sección 1.4.5). Se puede declarar un operador binario para II incluyendo `\DeclareSymbolFont{MiFuente}{OMS}{cmsy}{m}{n}` y `\DeclareMathSymbol{\MiAmalg}{\mathbin}{MiFuente}{161}` en el preámbulo del documento. Después de estas declaraciones, el comando `\MiAmalg` funciona como un operador binario que produce II. Véase la sección PARA SABER MÁS de este capítulo, donde encontrará referencias que le ayudarán a entender las líneas de código anteriores.

Cuadro 5.5: Símbolos para operadores binarios. Se dispone de los siguientes sinónimos: $\wedge \backslash land$, $\vee \backslash lor$, $\bowtie \backslash doublecup$, $\bowtie \backslash doublecap$

rrectamente a todos los cambios de fuente son las letras latinas mayúsculas. De hecho, *casi* todos los símbolos matemáticos que no son letras latinas no sufren modificaciones al cambiar de fuente. A pesar de que las letras latinas minúsculas, letras griegas mayúsculas y números responden adecuadamente para algunos cambios de fuentes, producen salidas extrañas para algunas otras, tal y como muestra el cuadro 5.12.

Las fuentes «blackboard bold» (`\mathbb`) y caligráficas (`\mathcal`) sólo están disponibles para las letras mayúsculas, mientras que las fuentes «fraktur» (`\mathfrak`) están disponibles tanto para mayúsculas como para minúsculas:

`\mathbb`: ABCDEFGHIJKLMNOPQRSTUVWXYZ.

`\mathcal`: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z.

\vee	$<$	\gnsim	$\not\leqslant$
$=$	$=$	\gtrapprox	$\not\lessdot$
\triangleright	$>$	\gtreqless	$\not\nprec$
\approx	\approx	\gtreqless	$\not\preceq$
\approx	\approx	\gtrless	\approx
\asymp		\gtrsim	\nsim
\backsim		\gvertneqq	\nsucc
\backsim		\leq	\prec
\backsim		\leqq	\precapprox
\backsim		\leqslant	\preccurlyeq
\circlearrowleft		\lessapprox	\preceq
\cong		\lesseqgtr	\precapprox
\curlyeqsucc		\lesseqgtr	\precneqq
\curlyeqsucc		\lessgtr	\precnsim
\doteq		\lessapprox	\precsim
\doteqdot		\ll	\risingdotseq
\eqcirc		\lll	\sim
\eqsim		\lnapprox	\simeq
\eqslantgr		\lneq	\succ
\eqslantless		\lneqq	\succapprox
\equiv		\lnsim	\succcurlyeq
\dots		\lvertneqq	\succceq
\fallingdotseq		\lvertneqq	\succnaprox
\geq		\ncong	\succnsim
\geqq		\neq	\succneqq
\geqslant		\ngeq	\succnsim
\gg		\ngeqq	\succsim
\ggg		\ngeqlant	\thickapprox
\gnapprox		\ngtr	\thicksim
\gneq		\nleq	\triangleq
\gneqq		\nleqq	\trianglelefteq

Cuadro 5.6: Símbolos para operadores de relación. Se dispone de los siguientes sinónimos: $\neq \backslash neq$, $\leq \backslash leq$, $\geq \backslash geq$, $\doteq \backslash Doteq$, $\ll \backslash llless$, $\gg \backslash ggtr$

$\mathfrak{mathfrak}$: $\mathfrak{A}\mathfrak{B}\mathfrak{C}\mathfrak{D}\mathfrak{E}\mathfrak{F}\mathfrak{G}\mathfrak{H}\mathfrak{I}\mathfrak{J}\mathfrak{K}\mathfrak{L}\mathfrak{M}\mathfrak{N}\mathfrak{O}\mathfrak{P}\mathfrak{Q}\mathfrak{R}\mathfrak{S}\mathfrak{T}\mathfrak{U}\mathfrak{V}\mathfrak{W}\mathfrak{X}\mathfrak{Y}\mathfrak{Z}$.
 $\mathfrak{a}\mathfrak{b}\mathfrak{c}\mathfrak{d}\mathfrak{e}\mathfrak{f}\mathfrak{g}\mathfrak{h}\mathfrak{i}\mathfrak{j}\mathfrak{k}\mathfrak{l}\mathfrak{m}\mathfrak{n}\mathfrak{o}\mathfrak{p}\mathfrak{q}\mathfrak{r}\mathfrak{s}\mathfrak{t}\mathfrak{u}\mathfrak{v}\mathfrak{w}\mathfrak{r}\mathfrak{y}\mathfrak{z}$.

Las fuentes «blackboard bold» y «fraktur» forman parte de la colección de fuentes que han sido compiladas por la *American Mathematical Society*, conocidas genéricamente como *AMS-Fonts*, y que nosotros estamos utilizando después de cargar el paquete *amssymb*. Las fuentes caligráficas

○	\circlearrowleft	≤	\Lleftarrow	↗	\nrightarrow
○	\circlearrowright	←	\longleftarrow	↖	\nwarrow
↶	\curvearrowleft	⇐	\Longleftarrow	→	\rightarrow
↷	\curvearrowright	↔	\longleftrightarrow	⇒	\Rrightarrow
⇓	\downdownarrows	⟱	\Longleftrightarrow	→	\rightarrowtail
↓	\downharpoonleft	⟵	\longmapsto	→	\rightharpoondown
↓	\downharpoonright	→	\longrightarrow	→	\rightharpoonup
↔	\hookleftarrow	⇒	\Longrightarrow	⇒	\rightleftarrows
↔	\hookrightarrow	⇒	\Longleftrightarrow	⇒	\rightleftharpoons
←	\leftarrow	↔	\looparrowleft	⇒	\rightrightarrows
←	\Leftarrow	⇒	\looparrowright	~~	\rightsquigarrow
←	\leftarrowtail	↑	\Lsh	⇒	\Rrightarrow
↑	\leftharpoondown	→	\mapsto	↑	\Rsh
↑	\leftharpoonup	→	\multimap	↙	\searrow
⤒	\leftleftarrows	≠	\nLeftarrow	↙	\swarrow
⤒	\leftrightarrow	≠	\nLeftrightarrow	⤒	\twoheadleftarrow
⤒	\Leftrightarrow	≠	\nrightarrow	⤒	\twoheadrightarrow
⤒	\leftrightsquigarrow	↗	\nearrow	↑	\upharpoonleft
⤒	\leftrightharpoons	↖	\nleftarrow	↑	\upharpoonright
⤒	\leftrightsquigarrow	↔	\nleftrightarrow	↑↑	\upuparrows

Cuadro 5.7: Símbolos para operadores de relación: flechas. Se dispone de los siguientes sinónimos: $\leftarrow \gets$, $\rightarrow \to$, \restriction

↑	\uparrow	↑	\Uparrow	↓	\downarrow
↓	\Downarrow	↓	\updownarrow	↔	\Updownarrow

Cuadro 5.8: Flechas verticales extensibles

que aparecen en la página 363 reciben el nombre `rsfs`, y no son las fuentes habituales. Las fuentes caligráficas habituales son:

`\mathcal`: *A B C D E F G H I J K L M N O P Q R S T U V W X Y Z*.

Nosotros hemos obtenido las fuentes `rsfs` porque estamos utilizando el paquete `mathptmx` (véase la página 185) y, para éste, las fuentes caligráficas en modo matemático son las `rsfs`. Alternativamente, el comando `\mathcal` también proporcionará las fuentes `rsfs` si se carga el paquete `calrsfs`, desarrollado por Ralph Smith, aunque no se tenga cargado el paquete `mathptmx`.

De forma adicional, el paquete `eucal`, que también forma parte de `AMSFONTS`, cambia la definición del comando `\mathcal` de modo que produce letras de tipo «Euler script» (en honor del matemático Leonhard Euler) *A*, *B*, *C*, etc., en lugar de caligráficas *A*, *B*, *C*, etc. Con la opción `mathscr` activa para `eucal` no se cambia la definición de `\mathcal`, y se dispone adicionalmente del comando

`\mathscr{SubFórmula}`

\exists	<code>\backepsilon</code>	\ntriangleright	\subsetneq
\because	<code>\because</code>	\ntrianglerighteq	\subsetneqq
\between	<code>\between</code>	\nvDash	\supset
\blacktriangleleft	<code>\blacktriangleleft</code>	\nVdash	\Supset
\blacktriangleright	<code>\blacktriangleright</code>	\nvDash	\supseteq
\bowtie	<code>\bowtie</code>	\nVDash	\supseteqq
\dashv	<code>\dashv</code>	\parallel	\supsetneq
\frown	<code>\frown</code>	\perp	\supsetneqq
\in	<code>\in</code>	\pitchfork	\therefore
\mid	<code>\mid</code>	\propto	\trianglelefteq
\models	<code>\models</code>	\shortmid	\trianglerighteq
\ni	<code>\ni</code>	\shortparallel	\varpropto
\nmid	<code>\nmid</code>	\smallfrown	\varsubsetneq
\notin	<code>\notin</code>	\smallsmile	\varsubsetneqq
\nparallel	<code>\nparallel</code>	\smile	\varsupsetneq
\nshortmid	<code>\nshortmid</code>	\sqsubset	\varsupsetneqq
\nshortparallel	<code>\nshortparallel</code>	\sqsubseteq	\vartriangle
\nsubseteq	<code>\nsubseteq</code>	\sqsupset	\vartriangleleft
\nsubseteqq	<code>\nsubseteqq</code>	\sqsupseteq	\vartriangleright
\nupseteq	<code>\nupseteq</code>	\subset	\vdash
\nupseteqq	<code>\nupseteqq</code>	\Subset	\Vdash
\ntriangleleft	<code>\ntriangleleft</code>	\subseteq	\vDash
\ntrianglelefteq	<code>\ntrianglelefteq</code>	\subseteqq	\VvDash

Cuadro 5.9: Símbolos para operadores de relación: miscelánea. Sinónimos: $\ni \owns$

que permite obtener letras de tipo «Euler script». El funcionamiento de `\mathscr` es análogo al comando `\mathcal`, y para utilizarlo debemos tener cargado el paquete `eucal` en el preámbulo de nuestro documento con la opción `mathscr`.

EJEMPLO 5.2

Caligráficas: $\mathcal{A}, \mathcal{B}, \mathcal{C}$
 Euler Script: $\mathscr{A}, \mathscr{B}, \mathscr{C}$

Caligráficas: $\mathcal{A}, \mathcal{B}, \mathcal{C}$
 Euler Script: $\mathscr{A}, \mathscr{B}, \mathscr{C}$

Fuentes matemáticas caligráficas: `\mathcal` y `\mathscr`

Hay situaciones en las que podemos necesitar incluir un determinado símbolo en negrita que no puede obtenerse mediante el comando `\mathbf`. Para estas situaciones podemos utilizar uno de los comandos siguientes:

`\boldsymbol{SubFórmula}`

`\pmb{SubFórmula}`

$\backslash\vert$	\backslashVert	/	/	$\backslash\backslash$	$\backslash\backslash$
$\backslash\arrowvert$	$\backslash\Arrowvert$,	$\backslash\bracevert$		

Cuadro 5.10: Símbolos extensibles no emparejados. La utilización de $\backslash\vert$, $\backslash\Vert$ o $\backslash\|$ como delimitadores emparejados no es aconsejable

(())	<	$\backslash\langle$	>	$\backslash\rangle$
[[]]	$\backslash\lceil$	$\backslash\lceil$	$\backslash\rceil$	$\backslash\rceil$
{	$\backslash\lbrace$	}	$\backslash\rbrace$	$\backslash\lfloor$	$\backslash\lfloor$	$\backslash\rfloor$	$\backslash\rfloor$
	$\backslash\lvert$		$\backslash\lvert$	$\backslash\lgroup$	$\backslash\lgroup$	$\backslash\rgroup$	$\backslash\rgroup$
$\backslash\ $	$\backslash\lVert$			$\backslash\lmoustache$	$\backslash\lmoustache$	$\backslash\rmoustache$	$\backslash\rmoustache$

Cuadro 5.11: Delimitadores de apertura y cierre

$\backslash\mathnormal$	X	x	0	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathbb$	\mathbb{X}	\curvearrowleft	\curvearrowright	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathbf$	\mathbf{X}	\mathbf{x}	$\mathbf{0}$	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathcal$	\mathcal{X}	\mathcal{x}	$\mathcal{0}$	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathfrak$	\mathfrak{X}	\mathfrak{x}	\mathfrak{o}	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathit$	X	x	$\mathit{0}$	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathrm$	X	x	$\mathrm{0}$	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathsf$	X	x	$\mathsf{0}$	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re
$\backslash\mathtt$	\mathtt{X}	\mathtt{x}	$\mathtt{0}$	$[]$	$+$	$-$	$=$	ξ	∞	\aleph	Σ	Π	\Re

Cuadro 5.12: Efecto de los cambios de fuente en modo matemático para letras latinas, griegas, números y símbolos

Los comandos $\boldsymbol{\text{boldsymbol}}$ y pmb afectan a todo lo que aparece en el argumento *SubFórmula*, excepto a las letras del alfabeto. El comando $\boldsymbol{\text{boldsymbol}}$ proporciona los símbolos en negrita, mientras que pmb , que se corresponde con el tipo de letra «poor man's bold», proporciona a cada símbolo el aspecto de estar en negrita, reescribiendo varias veces este símbolo con pequeños desplazamientos. Ésta es la forma de obtener la negrita con alguno de los símbolos matemáticos cuando $\boldsymbol{\text{boldsymbol}}$ no funciona. Hemos de hacer notar que $\boldsymbol{\text{boldsymbol}}$ no funciona con la versión actual del paquete *mathptmx*.

EJEMPLO 5.3

$\backslash[\backslash\lim\backslash\varphi=\backslash\mathbf{v}\backslash]$	$\lim\varphi = \mathbf{v}$
$\backslash[\backslash\boldsymbol{\text{boldsymbol}}\{\backslash\lim\backslash\varphi\}=\backslash\mathbf{v}\backslash]$	$\lim\varphi = \mathbf{v}$
$\backslash[\backslash\lim\backslash\varphi=\backslash\mathbf{v}\backslash]$	$\lim\varphi = \mathbf{v}$
$\backslash[\backslash\boldsymbol{\text{boldsymbol}}\{\backslash\lim\backslash\varphi\}=\backslash\mathbf{v}\backslash]$	$\lim\varphi = \mathbf{v}$
$\backslash[\backslash\text{pmb}\{\backslash\oint f\}=\backslash\boldsymbol{\text{boldsymbol}}\{\backslash\oint f\}\backslash]$	$\oint f = \boldsymbol{\text{boldsymbol}}\{\backslash\oint f\}$

Negritas para símbolos: $\boldsymbol{\text{boldsymbol}}$ y pmb . Las líneas 3 y 4 han sido compiladas utilizando las fuentes Computer Modern, mientras que las líneas 1, 2 y 5 han sido compiladas con las fuentes del paquete *mathptmx*

\acute{x}	$\acute{\tilde{x}}$	\dot{x}	\check{x}
\grave{x}	\widetilde{x}	\ddot{x}	\mathring{x}
\bar{x}	\widehat{x}	\dddot{x}	\vec{x}
\breve{x}	$\widehat{\widetilde{x}}$	$\ddot{\ddot{x}}$	

Cuadro 5.13: Acentos en modo matemático

5.5. Miscelánea matemática

ESTA sección está dedicada a comentar varias cuestiones que aparecen (o pueden aparecer) a la hora de escribir una fórmula, tales como el uso de acentos, modificaciones del tamaño de los símbolos, utilización de marcos, delimitadores de tamaño variable, etc.

5.5.1. Acentos en modo matemático

Tal y como se ha comentado en el apartado I.16.1, al escribir en modo matemático no se pueden utilizar los acentos de la misma forma en la que son empleados en modo ordinario. A la hora de escribir letras acentuadas en modo matemático hay que utilizar los comandos específicos que se listan a continuación:

$\acute{\text{Letra}}$	$\ddot{\text{Letra}}$	$\mathring{\text{Letra}}$
$\bar{\text{Letra}}$	$\ddot{\text{Letra}}$	$\tilde{\text{Letra}}$
$\breve{\text{Letra}}$	$\ddot{\text{Letra}}$	$\vec{\text{Letra}}$
$\check{\text{Letra}}$	$\grave{\text{Letra}}$	$\widehat{\text{Letra}}$
$\dot{\text{Letra}}$	$\hat{\text{Letra}}$	$\widetilde{\text{Letra}}$

El cuadro 5.13 muestra los diversos ejemplos de las letras acentuadas en modo matemático. Los comandos \widehat y \widetilde producen los mismos acentos que los comandos \hat y \tilde , pero su tamaño se adecua al del argumento. Es posible anidar los acentos en la forma $\hat{\hat{\text{Letra}}}$, $\ddot{\ddot{\text{Letra}}}$, etc., como se indica en el ejemplo 5.4, así como cambiar el tipo de fuente en la letra acentuada.

EJEMPLO 5.4

```
\[ \widehat{1+x} \neq \widehat{-y} \]
No hay puntos en $ \vec{\imath} + \vec{\mathfrak{j}} $.
Es posible utilizar acentos dobles como
$ \hat{\hat{\text{A}}} $, $ \mathring{\mathring{\text{A}}} $.
y cambiar la fuente y luego utilizar
acentos $ \hat{\varSigma} $, $ \dot{\mathbf{X}} $.
```

$$\widehat{1+x} \neq \widehat{-y}$$

No hay puntos en $\vec{i} + \vec{j}$.

Es posible utilizar acentos dobles como $\hat{\hat{A}}$ y cambiar la fuente y luego utilizar acentos $\hat{\Sigma}, \dot{\mathbf{X}}$.

Acentos en modo matemático. Cuando se tiene cargado el paquete `mathptmx` en su versión actual, `\jmath` no está disponible. La salida `j` corresponde al carácter '174 de la fuente `cmmi` con la codificación «OML». Se puede declarar un símbolo ordinario para `j` incluyendo `\DeclareSymbolFont{MiFuente}{OML}{cmmi}{m}{n}` y `\DeclareMathSymbol{\MiJmath}{\mathord}{MiFuente}{'174}` en el preámbulo del documento. Después de estas declaraciones, `\MiJmath` funciona como un símbolo ordinario que produce `j`. Véase la sección PARA SABER MÁS de este capítulo, donde encontrará referencias que le ayudarán a entender las líneas de código anteriores.

Los comandos

<code>\imath</code>	<code>\jmath</code>
---------------------	---------------------

producen las letras *i* y *j* sin el punto, preparadas para soportar el peso de un nuevo acento. El comando `\jmath` no está disponible con la versión actual del paquete *mathptmx*.

El paquete *accents*, diseñado por Javier Bezos [2], proporciona comandos que pueden ser muy útiles para definir y utilizar nuevos acentos en modo matemático, para agruparlos correctamente y para ponerlos debajo de los símbolos principales.

5.5.2. Puntos suspensivos en matemáticas

Los puntos suspensivos, que es el signo ortográfico de la elipsis gramatical (...), se usan tanto dentro como fuera de las fórmulas matemáticas. Para escribir fórmulas matemáticas se requieren, además de los normales, puntos suspensivos en vertical, en diagonal y puntos suspensivos horizontales un poco más elevados. Los comandos que siguen, cuya salida se encuentra en el cuadro 5.14, proporcionan estos signos:

<code>\dots</code>	<code>\vdots</code>	<code>\dotsc</code>	<code>\dotsi</code>
<code>\ldots</code>	<code>\ddots</code>	<code>\dotsb</code>	<code>\dotso</code>
<code>\cdots</code>		<code>\dotsm</code>	

No hay un consenso general sobre cuál es la forma de utilizar los puntos suspensivos en matemáticas; la decisión de cómo éstos se deben utilizar depende del contexto y puede ser una cuestión de gusto. El comando `\dots`, utilizado en modo matemático, produce los puntos suspensivos, o en la línea base, o en una posición centrada verticalmente (ligeramente elevada respecto de la línea base), dependiendo de lo que se escriba a continuación: si el siguiente carácter es un signo +, -, × (`\times`), etc., los puntos suspensivos se centrarán; si el siguiente carácter es una coma, los puntos suspensivos se pondrán sobre la línea base; si `\dots` se encuentra al final de una fórmula y le sigue algo como `\end`, \$, etc., no se tiene información de cómo colocar los puntos suspensivos y, en este caso, cuando se quiera modificar el comportamiento por defecto de `\dots`, debemos utilizar los comandos `\ldots` y `\cdots` que producen los puntos suspensivos sobre la línea base y en una posición un poco elevada (centrados), respectivamente. En lugar de `\ldots` o `\cdots`, la utilización de puntos suspensivos en fórmulas en nuestro documento puede adaptarse a diferentes gustos y convenios usando los comandos más especializados que siguen:

`\dotsc` Puntos suspensivos utilizados normalmente para omitir términos que se escriben separados por comas («`\dots with commas`»).

`\dotsb` Puntos suspensivos que sustituyen los términos intermedios de una sucesión en la que intervienen operadores de relación binaria («`\dots with binary operators`»).

`\dotsm` Puntos suspensivos que representan la elipsis de los términos intermedios de una sucesión de multiplicaciones implícitas («`\dots with multiplications`»).

`\dotsi` Puntos suspensivos a continuación del signo integral («`\dots with integrals`»).

`\dotso` Otros puntos suspensivos (ninguno de los de arriba).

.	.	:	\colon	\ldots	\ldots	\ldots	\ldots
/	/	:	:	\cdots	\cdots	\cdots	\cdots
		!	!	\dotsb	\dotsb	\ddots	\ddots
,	,	?	?	\dotsc	\dotsc	\vdots	\vdots
;	;	\dots	\dots	\dotsi	\dotsi		

Cuadro 5.14: Signos de puntuación

EJEMPLO 5.5

```
... sucesiones $a_{\{0\}},a_{\{1\}}\dots,a_{\{n\}},\ldots$ y $b_{\{0\}},b_{\{1\}}\dots,b_{\{n\}},\ldots$ el...
... convolución es
\[
    c_{\{k\}}:= a_{\{0\}}b_{\{k\}}+a_{\{1\}}b_{\{k-1\}}+
    \dots+a_{\{k\}}b_{\{0\}}
\]
... escribimos:
\[
    \int_{a_{\{1\}}}^{+\infty} \dotsi
\]
```

Dadas las sucesiones $a_0, a_1, \dots, a_n, \dots$ y $b_0, b_1, \dots, b_n, \dots$ el término k -ésimo de su producto de convolución es

$$c_k := a_0 b_k + a_1 b_{k-1} + \dots + a_k b_0$$

Para fórmulas integrales escribimos:

$$\int_{a_1}^{+\infty} \dots$$

**5.5.3. Guiones que no parten expresiones**

Se dispone del comando

\nobreakdash

para evitar la ruptura de una línea en el guión que le sigue. Por ejemplo, si se escribe «páginas 123–129» como páginas 123\nobreakdash–129, la ruptura de una línea nunca se producirá entre el guión – y 129. Se puede utilizar también \nobreakdash para prevenir rupturas en guiones de expresiones como « n -dimensional» escribiendo \$n\$\nobreakdash-dimensional.

5.5.4. Tamaños de las fórmulas

Al usar los comandos \normalsize, \large, etc. para seleccionar el tamaño de las fuentes, estamos estableciendo también el tamaño de las fórmulas: véanse los ejemplos I.16.5 y 5.6.

EJEMPLO 5.6

```
\[ \Sigma := \int_0^1 f(x) dx
\Large \[ \Sigma := \int_0^1 f(x) dx \]
```

$$\Sigma := \int_0^1 f(x) dx$$

$$\Sigma := \int_0^1 f(x) dx$$



Tamaño de las fuentes en modo matemático

Estos comandos para modificar el tamaño de las fuentes no funcionan dentro del modo matemático, por lo que no se pueden utilizar para establecer el tamaño de sólo una parte de la fórmula. Por otra parte, el tamaño de algunas expresiones (como las fracciones) y algunos símbolos que aparecen en una fórmula depende de la posición que ocupen en la misma y de que ésta esté resaltada (centrada) o forme parte de un párrafo. Esta dependencia está determinada por estilos que L^AT_EX tiene predefinidos, que son:

`displaystyle` predefinido para las fórmulas resaltadas (centradas);
`textstyle` predefinido para las fórmulas que se encuentran dentro de un párrafo;
`scriptstyle` predefinido para el primer nivel de subíndices y superíndices;
`scriptscriptstyle` predefinido para los demás niveles de subíndices y superíndices.

Si el estilo predefinido para una fórmula no nos satisface, podemos cambiarlo utilizando las declaraciones

<code>\displaystyle</code> <code>\scriptstyle</code>	<code>\textstyle</code> <code>\scriptscriptstyle</code>
---	--

El ejemplo I.16.11 muestra cómo utilizar `\displaystyle` y `\textstyle`. De forma similar se pueden utilizar los otros dos estilos. En el siguiente ejemplo mostramos la utilización de los comandos `\scriptstyle` y `\scriptscriptstyle`.

EJEMPLO 5.7

```
\[
f(a+h)=f(a)+\frac{f'(a)h}{1}+\textstyle
\frac{f''(a)h^2}{2!}+\dots+\frac{f^{(n)}(a)h^n}{n!}+o(h^n)
\frac{f''(a)h^2}{2!}+\dots+\frac{f^{(n)}(a)h^n}{n!}+o(h^n)
\frac{f^{(n)}(a)h^n}{n!}+o(h^n)
```

$$f(a+h) = f(a) + \frac{f'(a)h}{1} + \frac{f''(a)h^2}{2!} + \cdots + \frac{f^{(n)}(a)h^n}{n!} + o(h^n)$$

5.5.5. Fórmulas enmarcadas

En la lección 18 hemos explicado cómo construir cajas para enmarcar partes de un texto con los comandos `\fbox` y `\framebox`. Estos dos comandos, que no funcionan en modo matemático, admiten como parte del argumento cualquier fórmula del modo matemático ordinario. No admiten, sin embargo, fórmulas del modo matemático resaltado (centradas), lo que es sin duda una limitación para situaciones como la que presentamos en el ejemplo 5.8.

El siguiente comando puede utilizarse para enmarcar cualquier parte de cualquier fórmula:

<code>\boxed{SubFórmula}</code>

Este comando funciona en modo matemático y enmarca la *SubFórmula* contenida en su argumento. Su funcionamiento es similar al de `\fbox`, con la diferencia natural de que su argumento se procesa en modo matemático.

EJEMPLO 5.8

El número π ha suscitado el interés de la humanidad desde tiempos remotos.

```
\boxed{\int_0^{+\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}}
```

$$\boxed{\int_0^{+\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}}$$

 Fórmulas enmarcadas

El número π ha suscitado el interés de la humanidad desde tiempos remotos.

$$\boxed{\int_0^{+\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}}$$

$$\boxed{\int_0^{+\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}}$$

En el ejemplo 5.9 mostramos cómo se pueden igualar las alturas de las dos cajas que intervienen en la segunda parte del ejemplo anterior. Los entornos `Eqnarray` y `Eqnarray*`, disponibles en el paquete `fancybox` descrito en la sección 3.5.2, permiten enmarcar fórmulas resaltadas, pero guardando la estructura del entorno `eqnarray` que describiremos en la sección siguiente.

5.5.6. Espacios en fórmulas

Los comandos de espaciado presentados en la página 68 y en el cuadro I.16.1 funcionan tanto en modo matemático como fuera de él y, como ya se comentó, son los comandos que debemos utilizar para producir espacios en blanco en fórmulas. Hay otros comandos que dejan un espacio igual a la altura y/o anchura de un fragmento de `LATEX` que se procesa como argumento de los mismos. Éstos son:

<code></code>	<code>\hphantom{FragmentoTexto}</code>	<code>\vphantom{FragmentoTexto}</code>
---------------------------------------	--	--

El comando `\phantom` (resp. `\hphantom`; `\vphantom`) deja una espacio de la anchura y altura (resp. anchura y altura nula; anchura nula y altura) de la caja que se necesita para colocar la compilación de *FragmentoTexto*. El ejemplo que sigue utiliza `\vphantom` para igualar la altura de las cajas que aparecen en el ejemplo 5.8.

EJEMPLO 5.9

```
\boxed{\int_0^{+\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}}
```

$$\boxed{\int_0^{+\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}}$$

 Uso de `\vphantom` para igualar altura de cajas

5.5.7. Unos símbolos sobre otros

Subrayar es el ejemplo más sencillo de colocar algo debajo de un símbolo. En `LATEX` esto se puede hacer mediante el comando `\underline`, que funciona también en modo matemático. El comando `\overline` produce una raya sobre su argumento.

<code>\underline{Objeto}</code>	<code>\overline{Objeto}</code>
<code>\underbrace{Objeto}</code>	<code>\overbrace{Objeto}</code>

De forma similar, `\underbrace` y `\overbrace` se encargan de poner llaves en lugar de rayas, debajo y encima, respectivamente, de los objetos matemáticos que figuran en sus argumentos. Si añadimos un subíndice (resp. superíndice) a `\underbrace` (resp. `\overbrace`), éste aparecerá en un segundo nivel, centrado bajo (resp. sobre) la llave.

EJEMPLO 5.10

<pre>Encima y debajo:\; \$\overline{z_{\{2\}} + z^{\{2\}}}\$ \quad \underline{z_{\{2\}} + z^{\{2\}}}</pre> <pre>Llaves:\; \\$\overbrace{x+ (y+z)^{\{2\}}+w} \quad \overbrace{x+\underbrace{(y+z)_{\{2\}} +w}^{\{4\}}}</pre>	<pre>Encima y debajo: \$\overbrace{\overbrace{z_2 + z^2}^4}</pre> <pre>Llaves: \$\overbrace{x+(y+z)^2+w}^4 \quad \overbrace{x+y+z+w}^2\$</pre>
--	---

Rayas y llaves debajo y encima de objetos

Al referirnos a los acentos en modo matemático hemos visto cómo los comandos `\widehat` y `\widetilde` permiten poner acentos circunflejos o tildes sobre expresiones de más de una letra. Otros elementos como las flechas también pueden ponerse sobre expresiones matemáticas, o viceversa, expresiones matemáticas sobre flechas. Los comandos que siguen realizan esta labor.

<code>\overleftarrow{Objeto}</code> <code>\underleftarrow{Objeto}</code> <code>\overleftrightarrow{Objeto}</code> <code>\xleftarrow[Debajo]{Encima}</code>	<code>\overrightarrow{Objeto}</code> <code>\underrightarrow{Objeto}</code> <code>\underleftrightarrow{Objeto}</code> <code>\xrightarrow[Debajo]{Encima}</code>
---	---

Los tres primeros comandos de cada columna colocan la flecha correspondiente debajo y encima de *Objeto*, tal y como ilustra el ejemplo que sigue. Los últimos, `\xleftarrow` y `\xrightarrow`, se ocupan de poner los objetos *Debajo* y *Encima* debajo y encima, respectivamente, de la flecha correspondiente, tal y como se ilustró también en el apartado I.16.17.

EJEMPLO 5.11

En general se tiene: $\overleftarrow{A \times B} \neq \overrightarrow{A \times B}$. De forma similar: $\underleftarrow{z+w} \neq \underrightarrow{z+w} \neq \overleftrightarrow{zw}$. Estas flechas son autoextensibles: $\xleftarrow[T]{a+b} \quad \xleftarrow[a+b+c+d]{T}$	En general se tiene: $\overleftarrow{A \times B} \neq \overrightarrow{A \times B}$. De forma similar: $\underleftarrow{z+w} \neq \underrightarrow{z+w} \neq \overleftrightarrow{zw}$. Estas flechas son autoextensibles: $\xleftarrow[T]{a+b} \quad \xleftarrow[a+b+c+d]{T}$
---	--

Flechas sobre objetos y viceversa

Los comandos que siguen, de propósito más general que los anteriores, sirven para colocar objetos sobre, debajo y a los lados de otros objetos.

<code>\stackrel{Encima}{RelaciónBinaria}</code> <code>\underset{Debajo}{Objeto}</code> <code>\sideset{ArgumentoIzquierda}{ArgumentoDerecha}{Operador}</code>	<code>\overset{Encima}{Objeto}</code>
--	---------------------------------------

\LaTeX proporciona el comando `\stackrel` para colocar el argumento *Encima* sobre la relación binaria *RelaciónBinaria*. Más en general, los comandos `\overset` y `\underset` pueden ser utilizados para colocar los símbolos *Encima* y *Debajo*, respectivamente, encima y debajo de *Objeto*.

El comando `\sideset` resuelve el problema de situar índices y superíndices adicionales a la izquierda y a la derecha de operadores de tamaño variable (véase el cuadro 5.18). El ejemplo 5.12 recoge situaciones en las que `\sideset` puede resultar útil, como, por ejemplo, para colocar una prima ' en un sumatorio.

EJEMPLO 5.12

Convenimos que

$\$\\mathbb{N}\\stackrel{\\mathrm{def}}{=} \\{1,2,\\dots\\}$. Convenimos que $N \\stackrel{\\mathrm{def}}{=} \\{1,2,\\dots\\}$.

Tanto monta $\\overset{A}{B}$ monta tanto
 $\\underset{B}{A}$

Tanto monta $\\overset{A}{B}$ monta tanto A_B

La suma $\\sideset{}{}{\\sum_{n< k} nE_n}$ es\\dots

La suma $\\sum'_{n< k} nE_n$ es...

Superíndices y subíndices a gogó:\\
 $\\sideset{_*}{_*}{\\sideset{_*}{_*}{\\prod_{j=1}^k}}$

Superíndices y subíndices a gogó: $\\prod_{j=1}^k$

Objetos sobre objetos en modo matemático

5.5.8. El comando `\smash`

El comando

`\smash[Argumento]{Objeto}`

mantiene los contenidos de la caja que contiene a *Objeto* pero anula su altura y profundidad. *Argumento* puede tomar los valores `t` y `b` para anular, respectivamente, la altura y la profundidad de la caja. Este comando resulta especialmente útil para hacer más consistente el aspecto de radicales adyacentes de distinto tamaño, tal y como muestra el ejemplo que sigue.

EJEMPLO 5.13

Una de radicales:

```
\[x:=(1/\sqrt{x_{ij}})\sqrt{1+y}\] \quad x:=\\frac{1}{\\sqrt{x_{ij}}}\\sqrt{1+y}
```

Una de radicales:

$$x := (1/\sqrt{x_{ij}})\sqrt{1+y} \quad x := (1/\sqrt{x_{ij}})\sqrt{1+y}$$

Funcionamiento del comando `\smash`

5.5.9. Mejoras en las raíces

En el apartado I.16.2 hemos aprendido el uso del comando `\sqrt[n]{Radicando}` para obtener el símbolo del radical. Algunas veces la posición del orden de la raíz no es brillante, como ocurre en $\sqrt[p]{n}$, donde p debería estar un poco más elevado con respecto al radical. Los comandos que siguen, que se usan dentro del argumento optativo de `\sqrt`, permiten modificar la posición del orden de la raíz.

`\leftroot{Desplazamiento}`

`\uproot{Desplazamiento}`

Desplazamiento es un número que indica el desplazamiento del índice del radical hacia la izquierda en el comando `\leftroot`, o hacia arriba en el comando `\uproot`. Si *Desplazamiento* es un número negativo, el orden de la raíz se desplaza hacia la derecha con el primer comando, o hacia abajo con el segundo.

EJEMPLO 5.14

```
\Large \[ \sqrt[p]{k}\quad
\sqrt[\leftroot{-2}]{k}\quad
\sqrt[\leftroot{-1}\uproot{1}]{k}\quad
\]
```

$$\sqrt[p]{k}$$

$$\sqrt[\leftroot{-2}]{k}$$

$$\sqrt[\leftroot{-1}\uproot{1}]{k}$$

5.5.10. Más fracciones

Todas las capacidades de los comandos `\frac`, `\binom` y sus derivados, estudiadas en la lección 16, pueden ser asumidas por el siguiente comando de uso múltiple:

```
\genfrac{\DelIzquierda}{\DelDerecha}{\Grosor}{\Tamaño}{\Numerador}{\Denominador}
```

Los seis argumentos de `\genfrac` tienen la siguiente función. Los dos primeros, *DelIzquierda* y *DelDerecha*, contienen, opcionalmente, sendos símbolos que aparecerán a la izquierda y a la derecha de la fracción. El argumento *Grosor* es el grosor de la raya de fracción (si es `0pt` la raya es invisible, y si se deja vacío el grosor es el normal). Por último, el argumento *Tamaño* es un número entero entre 0 y 3 que se utiliza para fijar el tamaño de la fracción: 0 para `\displaystyle`, 1 para `\textstyle`, 2 para `\scriptstyle` y 3 para `\scriptscriptstyle` (véase la sección 5.5.4); si se deja este argumento vacío, el tamaño será el correspondiente a la fórmula de la que forme parte. La siguiente línea muestra cómo está definido el comando `\tbinom`:

```
\newcommand*{\tbinom}[2]{\genfrac{}{}{0pt}{}{#1}{#2}}
```

De la familia de las fracciones, pero de naturaleza distinta, son las fracciones continuas, que se escriben con el comando

```
\cfrac{Posición}{\Numerador}{\Denominador}
```

Las fracciones continuas escritas con este comando presentan mejor aspecto que las que se obtienen con el uso repetido de `\frac`. El argumento opcional *Posición* puede tomar los valores `l` o `r`, y sirve para que el numerador *Numerador* aparezca justificado, respectivamente, a la izquierda o a la derecha de la raya de fracción.

EJEMPLO 5.15

```
\[
\cfrac{l}{\sqrt{2}+\cfrac{2}{\sqrt{2}+\cfrac{3}{\sqrt{2}+\dots}}}
\]
```

$$\cfrac{1}{\sqrt{2}+\cfrac{2}{\sqrt{2}+\cfrac{3}{\sqrt{2}+\dots}}}$$

Fracciones continuas

5.5.11. Delimitadores: paréntesis, corchetes y llaves

Un delimitador es cualquier símbolo que actúa lógicamente como los paréntesis, los corchetes, las llaves, etc. (véase el cuadro 5.11). Normalmente, estos símbolos se utilizan de dos en dos para encerrar o *delimitar* una determinada expresión. L^AT_EX distingue, por ejemplo, entre los dos paréntesis a la hora de situarlos junto a la expresión que encierran: «(» es un delimitador de apertura y se sitúa más cerca de la expresión que tiene a su derecha que de la que queda a su izquierda, mientras que «)» lo hace al revés, tal y como muestra el ejemplo que sigue.

EJEMPLO 5.16

```
Asociatividad... $a+(b+c)=(a+b)+c$  
Se... \[a+(\frac{b}{c}+\frac{d}{e})=\frac{ace+be+cd}{ce}\]
```

Asociatividad de la suma: $a + (b + c) = (a + b) + c$
Se tiene la igualdad:

$$a + \left(\frac{b}{c} + \frac{d}{e}\right) = \frac{ace + be + cd}{ce}$$

Delimitadores: espacios en blanco y tamaño

Obsérvese que, en el ejemplo anterior, el uso de los paréntesis produce siempre un resultado satisfactorio en cuanto a su ubicación, pero no siempre en cuanto a su tamaño. Es de esperar que cuando encerramos una fórmula entre delimitadores, L^AT_EX elija el tamaño de éstos en función del tamaño de la fórmula. Esto es lo que ocurre cuando utilizamos los delimitadores en combinación con los comandos:

```
\leftDelimitadorIzquierda  
Fórmula  
\rightDelimitadorDerecha
```

Si se quiere poner un solo delimitador a la derecha o a la izquierda, se debe completar la pareja de comandos que delimitan la fórmula con `\left.` o `\right.`, respectivamente. También es posible utilizar un tipo de delimitador de apertura y uno distinto de cierre, situación que aparece, por ejemplo, al escribir intervalos semiabiertos de números reales.

EJEMPLO 5.17

Se tiene la desigualdad:

$$\left| \int_0^1 f(x) dx \right| \leq \int_0^1 |f(x)| dx$$

La función

$$\begin{array}{l} \left[f(x) = \left\{ \begin{array}{ll} \sin x & \text{si } x \neq 0 \\ 1 & \text{si } x = 0 \end{array} \right. \right] \\ \text{es continua.} \end{array}$$

Un intervalo semiabierto $\left(\frac{a}{b}, \frac{c}{d} \right]$...

Se tiene la desigualdad:

$$\left| \int_0^1 f(x) dx \right| \leq \int_0^1 |f(x)| dx$$

La función

$$f(x) = \begin{cases} \frac{\sin x}{x} & \text{si } x \neq 0 \\ 1 & \text{si } x = 0 \end{cases}$$

es continua.

Un intervalo semiabierto $\left(\frac{a}{b}, \frac{c}{d} \right]$...

Fórmulas, delimitadores. Véase la sección 5.6 para la descripción del entorno `array` y la sección 5.8 para la sintaxis de `\sin` y similares. La forma de presentar la definición de la función f se puede automatizar con el entorno `cases` que se introduce en la sección 5.7.6

La elección automática del tamaño de los delimitadores al utilizarlos conjuntamente con los comandos `\left` y `\right` tiene algunas limitaciones. Primero, el tamaño dado al delimitador es el necesario para abarcar el ítem más grande de la fórmula (esto puede resultar estéticamente lamentable en algunos casos: véanse los sumatorios del ejemplo anterior). Segundo, el escalado de los tamaños de los delimitadores no es continuo, y pequeñas variaciones en el tamaño de una fórmula pueden ocasionar grandes variaciones en el tamaño de los delimitadores. Estos inconvenientes aconsejan que a veces sea mejor elegir el tamaño de los delimitadores manualmente, lo que se puede hacer con los comandos

<code>\biglDelimitadorIzquierda</code>	<code>\bigrDelimitadorDerecha</code>
<code>\bigglDelimitadorIzquierda</code>	<code>\biggrDelimitadorDerecha</code>
<code>\BiglDelimitadorIzquierda</code>	<code>\BigrDelimitadorDerecha</code>
<code>\BigglDelimitadorIzquierda</code>	<code>\BiggrDelimitadorDerecha</code>

que producen delimitadores perfectamente escalados para los distintos tamaños de las fuentes de L^AT_EX. El cuadro 5.15 y el ejemplo que sigue muestran el efecto y utilidad de estos comandos. En el cuadro referido, todos los paréntesis que rodean la a son de tamaño normal y los que rodean la fracción $\frac{b}{c}$ están afectados por los modificadores de tamaño que aparecen en la columna correspondiente; de esta forma, puede verse de manera progresiva el efecto que produce la utilización de los comandos anteriores.

EJEMPLO 5.18

Obsérvese

```
\[\biggl\lvert\sum_{j=1}^{\infty}x_j\biggr\rvert
\biggr\lvert\sum_{j=1}^{\infty}\bigl\lvert x_j\bigr\rvert\biggr\rvert
\left\lvert\sum_{j=1}^{\infty}\left\lvert x_j\right\rvert\right\rvert
en contraposición a
\[\left\lvert\sum_{j=1}^{\infty}x_j\right\rvert
\left\lvert\sum_{j=1}^{\infty}\left\lvert x_j\right\rvert\right\rvert
\left\lvert\sum_{j=1}^{\infty}\left\lvert x_j\right\rvert\right\rvert]
```

Podemos conseguir más claridad cuando hay muchos paréntesis:

```
\[\biggl((x_1y_1)+(x_2y_2)\biggr)
\biggl((v_1w_1)+(v_2w_2)\biggr)\]
```

Tamaño de los delimitadores: escalado manual

Obsérvese

$$\left| \sum_{j=1}^{\infty} x_j \right| \leq \sum_{j=1}^{\infty} |x_j|$$

en contraposición a

$$\left| \sum_{j=1}^{\infty} x_j \right| \leq \sum_{j=1}^{\infty} |x_j|$$

Podemos conseguir más claridad cuando hay muchos paréntesis:

$$((x_1y_1)+(x_2y_2))((v_1w_1)+(v_2w_2))$$

Hemos de hacer notar que los delimitadores `\lvert`, `\rvert`, `\lVert` y `\rVert` han sido diseñados para *aminorar* el uso del símbolo `|`. Este símbolo es utilizado habitualmente en L^AT_EX para representar una gran variedad de objetos matemáticos como: divisiones $(p|q)$, restricción de aplicaciones $(f|_A)$, valor absoluto $(|x|)$, etc. El abuso de `|` puede conducir a que se esté dando el mismo tratamiento tipográfico a un símbolo cuyo tratamiento depende del contexto en el que se usa. Es aconsejable no utilizar `|` ni `\lvert` como delimitadores, porque la distribución de los espacios circundantes no es equivalente al espacio proporcionado por `\lvert`, `\rvert`, `\lVert` y `\rVert`.

Tamaño	<code>\left</code>	<code>\bigl</code>	<code>\Bigl</code>	<code>\biggl</code>	<code>\Biggl</code>
texto	<code>\right</code>	<code>\bigr</code>	<code>\Bigr</code>	<code>\biggr</code>	<code>\Biggr</code>
$(a)\left(\frac{b}{c}\right)$	$(a)\left(\frac{b}{c}\right)$	$(a)\bigl(\frac{b}{c}\bigr)$	$(a)\Bigl(\frac{b}{c}\Bigr)$	$(a)\biggl(\frac{b}{c}\biggr)$	$(a)\Biggl(\frac{b}{c}\Biggr)$

Cuadro 5.15: Comandos para escalado manual de los delimitadores

5.6. Matrices y determinantes

AS matrices y los determinantes son, en cierto sentido, como las tablas de texto descritas en la lección 14, y tienen un tratamiento similar. La estructura básica de L^AT_EX para estas construcciones matemáticas es el entorno `array`. En esta sección se describe dicho entorno a la vez que algunos otros entornos más específicos que incorporan por defecto delimitadores. Cuando en una tabla conviven fórmulas y texto, lo aconsejable es utilizar el paquete `array` (véase la sección 3.3.2).

El entorno `array`

El entorno `array` se utiliza en modo matemático para producir matrices, de forma similar a como el entorno `tabular` se utiliza en modo ordinario para producir tablas. Sus argumentos y opciones coinciden con los de `tabular`, los cuales pueden consultarse en la lección 14.

```
\begin{array}[Posición]{FormatoCol}
Elemento11 & Elemento12... & Elemento1N \\
Elemento21 & Elemento22... & Elemento2N \\
...
\end{array}
```

Tal y como muestra el siguiente ejemplo, la utilización de este entorno es totalmente similar a la del entorno `tabular`: obsérvese que `array` no inicia el modo matemático por sí mismo; hemos de iniciarlos nosotros.

EJEMPLO 5.19

```
\[\begin{array}{crl}
x & \& 3 & \& m+n^2 \\
x+y & \& 5 & \& m-n \\
x^{z} & \& \sqrt{75} & \& \\
(x+y)z' & \& 100 & \& 1+m
\end{array}\]
```



El entorno `array` puede utilizarse tanto en fórmulas resaltadas (centradas) como en fórmulas que son parte de un párrafo, y se pueden utilizar con él todos los delimitadores del cuadro 5.11, como ya se puso de manifiesto en el ejemplo 5.17. En el ejemplo siguiente escribimos una matriz utilizando el entorno `array` en combinación con los comandos `\left(` y `\right)`. El determinante se obtiene con el concurso de los comandos `\left\langle` `\right\rangle` y `\left|` `\right|`.

EJEMPLO 5.20

```
\[
\left( \begin{array}{lcl}
x & \left( \begin{array}{cc}
2 & 3 \\
1 & 4
\end{array} \right) & m+n^2 \\
x+y & \left( \begin{array}{c}
5 \\
m-n
\end{array} \right) & \\
x^z & \left( \begin{array}{c}
\sqrt{7} \\
m
\end{array} \right) & \\
yz' & \left( \begin{array}{c}
10 \\
1+m
\end{array} \right) &
\end{array} \right)
\]
```

$$\left(\begin{array}{ccc|c}
x & 2 & 3 & m+n^2 \\
& 1 & 4 & \\
x+y & 5 & & m-n \\
x^z & \sqrt{7} & & m \\
yz' & 10 & & 1+m
\end{array} \right)$$

Tal y como ya hemos detallado en la sección 3.3.2, el paquete `array`, distribuido con L^AT_EX y creado por Frank Mittelbach y David Carlisle [47], aporta mejoras en el manejo de los entornos `array` y `tabular`, proporcionando más tipos de columnas. El paquete `delarray` (delimiter-array) es una extensión del anterior y permite integrar los delimitadores en las opciones del entorno `array` (véase [21, 5.3.6]).

Entornos para matrices que ya incluyen delimitadores

Existen entornos específicos para escribir matrices y determinantes que ya incluyen por defecto los delimitadores correspondientes. Los entornos `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix` y `Vmatrix` integran por defecto los delimitadores (), [], {}, || y |||, respectivamente. Por razones de consistencia de nombres y estructura también se define un entorno `matrix` sin delimitadores. Para utilizar estos entornos se escribe

```
\begin{?matrix}
Elemento11 & Elemento12... & Elemento1N \\
Elemento21 & Elemento22... & Elemento2N \\
...
\end{?matrix}
```

sustituyendo ? por la letra correspondiente al delimitador. El argumento *ElementoIJ* representa cada uno de los elementos de la matriz. Los elementos de cada columna están separados por el símbolo & y, por defecto, se pueden utilizar hasta 10 columnas centradas. No es posible especificar la alineación de las columnas a izquierda o derecha como en el entorno `array`. El número máximo de columnas está determinado por el contador `MaxMatrixCols`, cuyo valor puede modificarse con los comandos correspondientes descritos en la lección 17. Al igual que con `tabular`, \\ (o \\[Salto]) inicia una nueva fila de la matriz. Dentro de estos entornos es posible utilizar el comando

```
\hdotsfor [Factor] {NúmeroColumnas}
```

que produce una fila de puntos en la matriz que abarca tantas columnas como se especifican en *NúmeroColumnas*. El argumento opcional *Factor* es un factor multiplicativo de la separación entre los puntos cuyo valor por defecto es 1.

EJEMPLO 5.21

```
\[
\begin{pmatrix}
a & b & c \\
d & e & f \\
\hdotsfor[.1]{3} \\
g & h & i
\end{pmatrix}
\]
```

$$\begin{pmatrix} a & b & c \\ d & e & f \\ \dots \\ g & h & i \end{pmatrix}$$

Puntos suspensivos en matrices

Si necesitamos escribir matrices como parte de un párrafo (no resaltadas), se puede utilizar el entorno `smallmatrix`

```
\begin{smallmatrix}
Elemento11 & Elemento12... & Elemento1N \\
Elemento21 & Elemento22... & Elemento2N \\
...
\end{smallmatrix}
```

para el que no existen delimitadores predeterminados. Este entorno produce matrices de tamaño reducido.

EJEMPLO 5.22

Insertar ... \$\\left(\\begin{smallmatrix}a&b\\\\c&d\\end{smallmatrix}\\right)\$... es sencillo.

Insertar matrices $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ en textos es sencillo.

Matrices en párrafos: matrices pequeñas

5.7. Fórmulas en varias líneas: alineamiento de ecuaciones

EN esta sección vamos a describir varios procedimientos para escribir fórmulas que, o bien por su tamaño o bien por su complejidad, requieren ser escritas en más de una línea. Son múltiples los entornos que se pueden utilizar para esto, con diferencias que van desde la numeración o no de las ecuaciones hasta la forma de alinear los bloques de las mismas. Recomendamos al lector que necesite alinear o escribir bloques de ecuaciones acudir primero a los ejemplos 5.23 y 5.24 para, visualmente, decidir qué entorno se adapta mejor a sus necesidades y después, sólo después, acudir a las explicaciones que damos sobre la sintaxis del entorno que le interese.

5.7.1. El entorno `eqnarray`

En ocasiones hay que escribir cadenas de fórmulas en más de una línea de forma que aparezcan centradas o resaltadas. Se parte de una fórmula inicial y se van haciendo transformaciones en ella hasta obtener una fórmula final. Cada una de esas fórmulas intermedias está separada de la anterior por un símbolo de relación (véase el cuadro 5.6). Para que las sucesivas transformaciones en la fórmula resulten más transparentes, y sea fácil seguir el proceso matemático subyacente, se

suele escribir alineando verticalmente los símbolos de relación que sirven de «frontera» entre una fórmula y su transformada en cada una de las líneas, por ejemplo:

$$\begin{aligned} (a+b)^2 - (a-b)^2 &= \\ (a^2 + 2ab + b^2) - (a^2 - 2ab + b^2) &= 4ab \end{aligned} \quad (5.1)$$

Ésta es una situación en la que hay tres columnas correspondientes, respectivamente, a la fórmula de partida, el separador = y la fórmula transformada. Por lo tanto, puede escribirse bajo la estructura general de un entorno `array`, con tres columnas en la posición `{rc1}`, situado entre `\[` y `\]`.

Como esta situación se presenta con cierta frecuencia y en ocasiones es necesario hacer referencia a alguna de las fórmulas intermedias, existe un entorno que conserva la estructura de `array` y que etiqueta con un número distinto cada una de las líneas que componen la fórmula, utilizando para ello el mismo contador que usa el entorno `equation` que se presentó en la lección 16 y en la sección 5.1. Se trata del entorno `eqnarray`, abreviatura de «`equation array`», que tiene la estructura siguiente

```
\begin{eqnarray}
FórmulaIzquierda1 & Separador1 & FórmulaDerecha1 \\
FórmulaIzquierda2 & Separador2 & FórmulaDerecha2 \\
...
\end{eqnarray}
```

donde *FórmulaIzquierdaI* es la fórmula que aparece a la izquierda, *SeparadorI* representa cada símbolo de separación y *FórmulaDerechaI* es la fórmula que aparece a la derecha. Cada salto de línea se indica con `\\\`, o sus variantes `\\\[Salto]` o `*` (véase el apartado I.3.1). Este entorno permite que `LATEX` pueda introducir saltos de página entre las líneas de las ecuaciones. La orden `*` para terminar una línea instruye a `LATEX` para que no termine la página en esta línea. Al igual que con el entorno `equation`, cuando se comienza un entorno `eqnarray` se entra directamente en modo matemático centrado. La ecuación (5.1) se ha obtenido como indica el ejemplo que sigue:

EJEMPLO 5.23

```
\begin{eqnarray}
(a+b)^2 - (a-b)^2 &= & (a+b)^2 - (a-b)^2 &= \\ 
(a^2+2ab+b^2) - (a^2-2ab+b^2) &= & (a^2+2ab+b^2) - (a^2-2ab+b^2) &= 4ab \\
&\& 4ab\text{\nonumber} \\
\end{eqnarray}
```

Una fórmula en varias líneas con `eqnarray`

Para no numerar una línea se usa (delante del comando `\\\` si lo hubiere) el comando

```
\nonumber
```

En presencia del paquete `amsmath`, que siempre suponemos cargado, el comando `\notag` tiene la misma misión que `\nonumber` (véase la página 382). Existe la versión `eqnarray*` con la misma sintaxis y capacidades que el entorno `eqnarray` y que deja sin numerar todas las líneas de las ecuaciones que intervienen.

5.7.2. Comparación de entornos para partir y alinear ecuaciones en varias líneas

Además del entorno `eqnarray`, disponemos de los siguientes entornos para escribir fórmulas resaltadas (centradas) en más de una línea:

<code>equation</code>	<code>equation*</code>	<code>align</code>	<code>align*</code>
<code>alignat</code>	<code>alignat*</code>	<code>flalign</code>	<code>flalign*</code>
<code>gather</code>	<code>gather*</code>	<code>multline</code>	<code>multline*</code>
<code>split</code>			

El entorno `split` es un entorno especial que tiene que ser utilizado como entorno subordinado dentro de los otros entornos. No puede utilizarse en combinación con `multline`.

El ejemplo 5.24 muestra las peculiaridades de los entornos anteriores. Preste atención al hecho de que algunos entornos sólo se utilizan para partir ecuaciones, otros para agruparlas y otros para, en su caso, alinearlas.

Brevemente, la descripción de estos entornos es la que figura a continuación.

Entornos para una ecuación en una o varias líneas: `equation` se utiliza para ecuaciones de una línea resaltada; en combinación con `split` se puede utilizar para ecuaciones que necesitan más de una línea, pudiendo marcar con el signo de tabulación & el lugar donde queremos alinear las distintas líneas; el entorno `multline` también se utiliza para una única ecuación que se puede partir en varias líneas cuya disposición es controlada por el propio entorno: primera línea justificada a la izquierda y última justificada a la derecha salvo por unos pequeños márgenes.

Dado que con estos entornos se está escribiendo una única ecuación, sólo se numera la ecuación, no las líneas. Las versiones con * no numeran la ecuación.

Entornos para varias ecuaciones en varias líneas: los entornos `gather`, `align`, `alignat` y `flalign` se utilizan para escribir varias ecuaciones que forman un grupo. El primero centra las ecuaciones, mientras que en los otros dos se puede marcar el punto de alineación de las ecuaciones, permitiendo alinear bloques entre sí. El último entorno ubica las fórmulas ocupando el ancho del área de impresión.

En estos entornos se numera cada ecuación (cada línea), salvo que estemos utilizando la versión con asterisco (*).

Cuando se escriben varias líneas de ecuaciones éstas se van separando unas de otras con el comando `\backslash\backslash`. Los comandos

<code>\notag</code>	<code>\tag{Texto}</code>	<code>\tag*{Texto}</code>
---------------------	--------------------------	---------------------------

se utilizan para modificar las etiquetas que aparecen al lado de las ecuaciones. El comando `\notag` sirve para suprimir la etiqueta de una línea: se debe incluir antes de `\backslash\backslash` tal y como aparece en el ejemplo 5.24, en la parte correspondiente al entorno `gather`. También se puede terminar una línea con el comando `\tag{Texto}`, que escribe una etiqueta en esta línea imprimiendo *Texto* entre paréntesis, en lugar del número de ecuación. El comando `\tag*` actúa como `\tag` suprimiendo los paréntesis que delimitan la etiqueta. Los comandos `\tag` y `\tag*` también se pueden usar en las versiones con * de los entornos.

EJEMPLO 5.24

```
\begin{equation*}
(a+b)^2= a^2+2ab+b^2
\end{equation*}
```

$$(a+b)^2 = a^2 + 2ab + b^2$$

```
\begin{equation}
(a+b)^2= a^2+2ab+b^2
\end{equation}
```

$$(a+b)^2 = a^2 + 2ab + b^2 \quad (5.2)$$

```
\begin{equation}\begin{aligned}
(a+b)^4 &= (a+b)^2(a+b)^2 \\
&= a^4+4a^3b+6a^2b^2 \\
&\quad +4ab^3+b^4\end{aligned}\end{equation}
```

$$\begin{aligned}
(a+b)^4 &= (a+b)^2(a+b)^2 \\
&= a^4+4a^3b+6a^2b^2 \\
&\quad +4ab^3+b^4
\end{aligned} \quad (5.3)$$

```
\begin{multline}
(a+b)^4 = (a+b)^2(a+b)^2 \\
= a^4+4a^3b+6a^2b^2 \\
+4ab^3+b^4\end{multline}
```

$$\begin{aligned}
(a+b)^4 &= (a+b)^2(a+b)^2 \\
&= a^4+4a^3b+6a^2b^2 \\
&\quad +4ab^3+b^4
\end{aligned} \quad (5.4)$$

```
\begin{gather}
(a+b)^4 = a^4+4a^3b+6a^2b^2+4ab^3+b^4 \\
+4ab^3+b^4\notag\\
(a+b)^2= a^2+2ab+b^2 \quad \end{gather}
```

$$\begin{aligned}
(a+b)^4 &= a^4+4a^3b+6a^2b^2+4ab^3+b^4 \\
(a+b)^2 &= a^2+2ab+b^2
\end{aligned} \quad (5.5)$$

```
\begin{aligned}
(a+b)^4 &= a^4+4a^3b+6a^2b^2+4ab^3+b^4 \\
(a+b)^2 &= a^2+2ab+b^2\end{aligned}\end{aligned}
```

$$\begin{aligned}
(a+b)^4 &= a^4+4a^3b+6a^2b^2+4ab^3+b^4 \\
(a+b)^2 &= a^2+2ab+b^2
\end{aligned} \quad (5.6) \quad (5.7)$$

```
\begin{aligned}
a_{11}&=b_{11} & a_{12}&=b_{12} \\
a_{21}&=b_{21} & a_{22}&=b_{22}+c_{22}\end{aligned}\end{aligned}
```

$$a_{11} = b_{11} \quad a_{12} = b_{12} \quad (5.8)$$

$$a_{21} = b_{21} \quad a_{22} = b_{22} + c_{22} \quad (5.9)$$

```
\begin{flalign*}
a_{11}&=b_{11}&a_{12}&=b_{12}\\
a_{21}&=b_{21}&a_{22}&=b_{22}+c_{22}\end{flalign*}
```

$$a_{11} = b_{11} \quad a_{12} = b_{12}$$

$$a_{21} = b_{21} \quad a_{22} = b_{22} + c_{22}$$

Comparación de los distintos entornos para ecuaciones. Las rayas verticales indican los márgenes de impresión, y han sido incluidas para dar una referencia de la ubicación de las fórmulas al utilizar cada uno de los entornos

La ubicación de las etiquetas en fórmulas con varias líneas de ecuaciones puede resultar muy compleja. Si la posición de un número no es la adecuada, se puede utilizar el comando

```
\raisebox{Longitud}{}
```

para desplazar el número en cuestión verticalmente la *Longitud* dada, hacia arriba si es positivo y hacia abajo si es negativo.

5.7.3. Fórmulas demasiado largas: los entornos `split` y `multiline`

Para partir («split») una sola ecuación demasiado larga en varias líneas y alinear distintas partes de la ecuación, disponemos del entorno `split`, cuya sintaxis es:

```
\begin{split}
Parte1 & Parte2 \\
Parte3 & Parte4 \\
...
\end{split}
```

Lo único que hay que señalar en cada línea del entorno es el punto de alineación, que se indica poniendo un tabulador `&`. El entorno `split` no inicia el modo matemático por sí mismo, por lo que sólo se puede utilizar dentro de una fórmula declarada con `\[\]` o con entornos `equation`. Si estamos utilizando `split` dentro de un entorno `equation`, se numerará la ecuación en la línea que esté predeterminada por las opciones del paquete `amsmath`: en el ejemplo 5.24 se puede observar que el número de la ecuación aparece centrado en relación con la fórmula. Ésta es la opción por defecto, `centertags` (véase la sección 5.2).

Otro entorno que permite partir en varias líneas una ecuación es `multiline`, cuya sintaxis es la siguiente:

```
\begin{multiline}
Línea1 \\
Línea2 \\
...
\end{multiline}
```

Las distintas líneas de la ecuación se separan con el salto de línea `\\"`. La primera línea aparece a la distancia prefijada `\multlinegap` del margen izquierdo y la última línea a la misma distancia del margen derecho. No obstante, estas longitudes se pueden modificar con los comandos `\setlength` y `\addtolength`, que aparecen descritos en la lección 17. Las demás líneas intermedias aparecen centradas, aunque también pueden ser desplazadas a la izquierda o a la derecha mediante los comandos:

<code>\shoveleft{Línea}\\"</code>	<code>\shoveright{Línea}\\"</code>
-----------------------------------	------------------------------------

donde el argumento `Línea` es toda la línea de la ecuación que queremos desplazar.

Al igual que con el entorno `equation`, al utilizar `multiline` se entra automáticamente en modo matemático y se asigna un solo número de ecuación al conjunto de líneas. El número de la ecuación se ubica en la última línea (si está activa la opción `reqno`) o en la primera línea (si está activa la opción `leqno`).

EJEMPLO 5.25

```
\begin{multiline} a+b+c\\
+d+e+f\\
+i+j+k\\
\shoveleft{+l+m+n}\\
+\tilde{n}+o+p+q\\
+r+s+t \end{multiline}
```

$$\begin{aligned}
 & a + b + c \\
 & + d + e + f \\
 & + i + j + k \\
 & + l + m + n \\
 & + \tilde{n} + o + p + q \\
 & r + s + t
 \end{aligned}
 \tag{5.10}$$

5.7.4. Varias ecuaciones en varias líneas: gather, align, flalign y alignat

El entorno `gather` se utiliza para escribir un grupo consecutivo de ecuaciones de forma que cada una de ellas aparezca centrada y numerada en una línea.

```
\begin{gather}
Ecuación1 \\
Ecuación2 \\
Ecuación3 \\
...
\end{gather}
```

En las ecuaciones de este entorno no se puede incluir ningún tabulador &. El ejemplo 5.24 ilustra su utilización, y en él se observa el empleo de `\notag` para evitar que se numere una de las ecuaciones que se han escrito con `\gather`.

El entorno `align` se utiliza para alinear verticalmente las diferentes ecuaciones. Normalmente se alinean los símbolos de relación binaria. A diferencia de `eqnarray`, con este entorno se puede fijar el número de columnas de ecuaciones de acuerdo con nuestras necesidades. Su estructura es la siguiente:

```
\begin{align}
Ecuación1_1 & Ecuación1_2 & ... & Ecuación1_N \\
Ecuación2_1 & Ecuación2_2 & ... & Ecuación2_N \\
Ecuación3_1 & Ecuación3_2 & ... & Ecuación3_N \\
...
\end{align}
```

donde cada ecuación consiste en *FórmulaIzquierda & Separador FórmulaDerecha*, y las distintas ecuaciones de una misma línea, si existen, se van separando con un tabulador &. Si en el entorno `align` hay varias columnas, éstas aparecerán centradas entre los márgenes de la página. Con el entorno `flalign` se consigue que las columnas de los lados aparezcan junto a los márgenes. El ejemplo 5.24 muestra la utilización de `align` y `flalign`.

La distancia entre columnas consecutivas de ecuaciones puede fijarse con la siguiente variante de `align`:

```
\begin{alignat}{NúmeroColumnasEcuaciones}
Ecuación1_1 & Longitud Ecuación1_2 & ... & Longitud Ecuación1_N \\
Ecuación2_1 & Ecuación2_2 & ... & Ecuación2_N \\
...
\end{alignat}
```

El argumento *NúmeroColumnasEcuaciones* es el número de columnas de ecuaciones. Las distancias entre las columnas se definen en la primera fila de ecuaciones junto a los tabuladores & que las separan. *Longitud* es un comando (por ejemplo: `\hspace{12pt}`, `\quad...`) que establece la distancia de separación entre la primera y la segunda columna.

EJEMPLO 5.26

```
\begin{alignedat}{2}
x &= y & a &= b+c \\
x' &= y' & a' &= b \\
x+x' &= y+y' & a'b &= c'b
\end{alignedat}
```

$$\begin{array}{lll} x = y & a = b + c & (5.11) \\ x' = y' & a' = b & (5.12) \\ x + x' = y + y' & a'b = c'b & (5.13) \end{array}$$


```
\begin{alignedat}{2}
x &= y & a &= b+c \\
x' &= y' & a' &= b \\
x+x' &= y+y' & a'b &= c'b
\end{alignedat}
```

$$\begin{array}{lll} x = y & a = b + c & (5.14) \\ x' = y' & a' = b & (5.15) \\ x + x' = y + y' & a'b = c'b & (5.16) \end{array}$$

Separación de columnas en el entorno alignat

5.7.5. Bloques de fórmulas

Los entornos `align`, `alignat` y `gather` han sido diseñados para producir estructuras de ecuaciones que ocupan todo el ancho del texto. Esto significa, por ejemplo, que uno no puede enfrentar bloques de ecuaciones. Las variantes `aligned`, `alignedat` y `gathered` ocupan una anchura que corresponde a la anchura total de su contenido. El siguiente ejemplo ha sido producido con `aligned` y `gathered`.

EJEMPLO 5.27

```
\begin{equation*}
\begin{aligned}
&\alpha = b+c \\
&\beta = \alpha + \gamma \\
&x = y
\end{aligned}
\qquad \text{frente a} \qquad
\begin{aligned}
&a = b+c \\
&\beta = \alpha + \gamma \qquad \text{frente a} \qquad e = f \times g \\
&x = y \qquad \qquad \qquad \eta = \eta \times \varphi \\
&w = z
\end{aligned}
\end{equation*}
```

Un bloque frente a otro con `aligned` y `gathered`

El entorno `aligned` funciona como `align`, pero además podemos ahora utilizar un parámetro opcional para ubicar el bloque verticalmente. La sintaxis es:

```
\begin{aligned}[Posición]
Ecuación11 & Ecuación12 & ... & Ecuación1N \\
Ecuación21 & Ecuación22 & ... & Ecuación2N \\
...
\end{aligned}
```

Por defecto, las cajas producidas aparecerán centradas verticalmente en la línea base de la fórmula a la que pertenecen. Con el argumento *Posición* es posible modificar su posición; `b` hace que la caja se sitúe sobre la línea de la fórmula, mientras que `t` hace que la caja se sitúe debajo de la misma.

La sintaxis de `alignedat` y `gathered` es a la sintaxis de `alignat` y `gather` lo que la sintaxis de `aligned` es a la de `align`: no la repetimos.

5.7.6. El entorno `cases`

Una construcción habitual en matemáticas es la de alinear distintos casos (ecuaciones) posibles agrupándolos, además, con una llave `{` a la izquierda (por ejemplo, para dar una definición con opciones). Esto se podría conseguir con el entorno `array` y el delimitador `\left\{` a su izquierda, tal y como ilustra el ejemplo 5.17. El entorno específico `cases` realiza esta tarea de forma más automatizada.

```
\begin{cases}
Fórmula1 & Caso1 \\
Fórmula2 & Caso2 \\
...
\end{cases}
```

EJEMPLO 5.28

```
\[
P_{r-j}=
\begin{cases}
0 & \text{si } r-j \text{ es impar,} \\
(-1)^{(r-j)/2} & \text{si } r-j \text{ es par.}
\end{cases}
```

$$P_{r-j} = \begin{cases} 0 & \text{si } r-j \text{ es impar,} \\ (-1)^{(r-j)/2} & \text{si } r-j \text{ es par.} \end{cases}$$

Una definición con opciones: el entorno `cases`

5.7.7. Incluyendo texto en grupos de ecuaciones alineadas

Es posible introducir pequeñas aclaraciones de texto o líneas enteras en medio de cualquiera de las filas de ecuaciones alineadas con `gather`, `align` y sus variantes, sin necesidad de abandonar el entorno en uso. Para ello disponemos de los comandos

<code>\intertext{Texto}</code>	<code>\text{Texto}</code>
--------------------------------	---------------------------

El primer comando tiene que ir tras `\backslash\backslash` y produce una línea que contiene el *Texto*. El segundo comando debe ir antes de `\backslash\backslash` y puede servir para una pequeña aclaración; también puede utilizarse en una fórmula sencilla, tal y como se expuso en el apartado I.16.3.

EJEMPLO 5.29

```
\begin{aligned*}
a_1 &= b+c \quad \& \\
a_2 &= c+d \quad \& \\
\text{por } \eqref{MiEjemploEqnarray} &\quad \\
a_3 &= d+e \quad \& \\
\intertext{y sin lugar a dudas} \\
a_4 &= e+f \quad \& \\
\text{por } \eqref{ej:ComparDistEntorEc} &\quad \\
\end{aligned*}
```

Incluyendo texto en ecuaciones alineadas: los comandos `\intertext` y `\text`

5.7.8. Saltos de página en fórmulas de varias líneas

Cuando se escriben varias líneas de ecuaciones con alguno de los entornos anteriores que no sea el entorno `eqnarray`, L^AT_EX no puede cambiar de página en una de las líneas que forman el grupo de ecuaciones. Para permitírselo hay que escribir el comando de L^AT_EX

```
\allowdisplaybreaks[n]
```

en el preámbulo del documento. El argumento opcional *n* puede tomar los valores de 1 a 4 con el significado siguiente: 1 significa que se permite el salto de página entre líneas de una fórmula, pero se debe evitar tanto como sea posible; los valores 2, 3 y 4 se utilizan para permisividad creciente. Cuando `\allowdisplaybreaks` ha sido activado, se puede utilizar `**` para indicar que no queremos que el salto de página se produzca en una determinada línea.

Aunque no se use `\allowdisplaybreaks`, se puede permitir un salto de página en una fórmula individual poniendo, justo antes del salto de línea `\`, donde queremos que tenga efecto, el comando:

```
\displaybreaks[n]
```

5.7.9. Numerando ecuaciones

Son varias las opciones para personalizar la numeración de ecuaciones que lleva a cabo el contador `equation`. Es conveniente señalar que, en la clase de documento `article`, la representación de este contador sólo contiene el número de la ecuación, mientras que en la clase de documento `book` el número del capítulo acompaña al de la ecuación y además este contador está subordinado al contador `chapter` de los capítulos. Se puede redefinir el comando `\theequation` que imprime la representación del contador con los comandos estudiados en la lección 17. El comando

```
\numberwithin{Contador1}{Contador2}
```

subordina *Contador1* a *Contador2*. En particular podemos utilizar este comando para subordinar el contador `equation` al contador que queramos.

Para numerar grupos de ecuaciones en forma distinta a como se hace globalmente en el documento, disponemos del entorno `subequations`

```
\begin{subequations}
ParteDelDocumento
\end{subequations}
```

Inmediatamente después de `\begin{subequations}` se pueden hacer los cambios que necesitemos sobre el contador `equation`, y éstos sólo afectarán a las ecuaciones que aparezcan antes de `\end{subequations}`. Si no efectuamos ninguna definición particular, la actuación de este entorno consiste en que todas las ecuaciones escritas en la *ParteDelDocumento* encerrada en el mismo se enumeran en la forma (6a) (6b) ... si la última ecuación antes de entrar en el entorno fue la (5), y al salir del entorno la siguiente ecuación será la (7). Si observamos las referencias cruzadas realizadas en el ejemplo 5.30 vemos cómo se puede referenciar el entorno `subequations` o alguna de sus ecuaciones y cuáles son las referencias obtenidas.

arccos	\arccos	det	\det	líminf	\liminf	sinh	\sinh
arcsin	\arcsin	dim	\dim	lím sup	\limsup	sup	\sup
arctan	\arctan	exp	\exp	ln	\ln	tan	\tan
arg	\arg	gcd	\gcd	log	\log	tanh	\tanh
cos	\cos	hom	\hom	máx	\max	lim	\varinjlim
cosh	\cosh	inf	\inf	mín	\min	lim	\varprojlim
cot	\cot	inj lim	\injlim	Pr	\Pr	lim	\varliminf
coth	\coth	ker	\ker	proj lim	\projlim	lim	\varlimsup
csc	\csc	lg	\lg	sec	\sec		
deg	\deg	lím	\lim	sin	\sin		

Cuadro 5.16: Nombres de funciones, véase la sección 5.9

EJEMPLO 5.30

```
\begin{subequations}\label{n=2}
\begin{equation}
(a+b)(a-b)=a^2-b^2
\end{equation}
En efecto: \begin{aligned} (a+b)(a-b) \\ &= a^2 +ab -ab -b^2 \label{desarr} \\ &= a^2 -b^2 \nonumber \end{aligned}
\end{subequations}
Para diferencias de cubos tenemos:
\begin{equation} (a^2 +ab+ b^2)(a-b)= a^3 -b^3 \label{n=3} \end{equation}
Se ha abordado el caso $n=2$ en \eqref{n=2}, su desarrollo en \eqref{desarr}, y el caso $n=3$ en \eqref{n=3}
```

Numerando ecuaciones y subecuaciones

$$(a+b)(a-b) = a^2 - b^2 \quad (5.17a)$$

En efecto:

$$\begin{aligned} (a+b)(a-b) &= a^2 +ab -ab -b^2 \\ &= a^2 -b^2 \end{aligned} \quad (5.17b)$$

Para diferencias de cubos tenemos:

$$(a^2 +ab+ b^2)(a-b) = a^3 -b^3 \quad (5.18)$$

Se ha abordado el caso $n = 2$ en (5.17), su desarrollo en (5.17b), y el caso $n = 3$ en (5.18)

5.8. Nombres de funciones

NA función genérica en matemáticas se designa, por ejemplo, por las letras f , g , etc., y, sin embargo, existen funciones especiales como \cos , \log y otras, que tienen un símbolo específico para designarlas. Estas funciones se escriben normalmente en una fuente de tipo «roman» para hacerlas visualmente distintas de las funciones que se escriben con una sola letra. El cuadro 5.16 contiene la lista de las funciones disponibles en L^AT_EX, cuyo nombre es la abreviación, en inglés, del de la función que se quiere representar. Como cualquier otra expresión matemática, los comandos relacionados en el cuadro 5.16 admiten subíndices y superíndices. Algunos de éstos, como por ejemplo \lim , \sup o \max , actúan de forma análoga a como lo hace \sum en cuanto a la posición de los índices con respecto al nombre de la función, según aparezcan en una fórmula centrada o en una fórmula incluida en un párrafo (véase el ejemplo 5.31).

Si estamos utilizando el paquete `babel` con la opción `spanish`, además de las funciones del cuadro 5.16 disponemos, adicionalmente, de las versiones en castellano que figuran en el cuadro 5.17.

sen	\sen	arc sen	\arcsen	tg	\tg	arc tg	\arctg
-----	------	---------	---------	----	-----	--------	--------

Cuadro 5.17: Funciones trigonométricas en castellano

A pesar de que ya disponemos de una larga lista de funciones predefinidas, puede que necesitamos utilizar alguna otra. Para definir una nueva función se dispone de los comandos

```
\DeclareMathOperator{ \NombreComando }{SímboloFunción}
\DeclareMathOperator*{ \NombreComando }{SímboloFunción}
```

donde el argumento `\NombreComando` es el nombre del comando que produce `SímboloFunción` con los adecuados tipo de letra y espacios a los lados. El argumento `SímboloFunción` aparecerá en un modo de falso-texto: por ejemplo, el signo menos y el asterisco aparecerán, respectivamente, como el guión ortográfico - y el asterisco elevado * de las fuentes de texto, en lugar del signo menos – y del asterisco centrado * del modo matemático. Por otra parte, el nombre del operador se procesa en modo matemático, por lo que se pueden utilizar, por ejemplo, subíndices y superíndices. La versión estrellada `\DeclareMathOperator*` se utiliza para definir operadores donde los subíndices y superíndices se comporten como en los comandos `\sum` o `\lim`, situándolos a la derecha si la fórmula forma parte de un párrafo o centrados si la fórmula aparece resaltada. Formalmente, estos comandos de declaración deben ir en el preámbulo del documento. Si incluimos allí la definición `\DeclareMathOperator*{\supesen}{\sup \, , \, esen}`, podremos compilar el ejemplo que sigue.

EJEMPLO 5.31

Mientras $\lim_{z \rightarrow 0} e^z = 1$ el límite
 $\lim_{z \rightarrow 0} p(z) e^{1/z}$
no existe en \mathbb{C}_{∞} ...

Definido $\lim_{z \rightarrow \infty} f(z) = \infty$
debe tenerse claro que $\sup_{x \in \mathbb{R}^n} |f(x)| < \infty$.

Mientras $\lim_{z \rightarrow 0} e^z = 1$ el límite
 $\lim_{z \rightarrow 0} p(z) e^{1/z}$
no existe en \mathbb{C}_{∞} ...

Definido $\|f\|_{\infty} = \sup_{x \in \mathbb{R}^n} |f(x)|$
debe tenerse claro que $\sup_{x \in \mathbb{R}^n} |f(x)| \neq \sup_{x \in \mathbb{R}^n} |f(x)|$.

Funciones: utilización y declaración

El comando `\mod`

Los comandos

\mod	\bmod	\pod	\pmod
------	-------	------	-------

se utilizan para la notación «mod» que se refiere a las congruencias algebraicas. En general todas estas versiones de `\mod` difieren en el tratamiento de los espacios y en que las versiones que empiezan por `p` incluyen un paréntesis mientras que las otras no lo hacen. La utilización de una u otra de estas versiones depende fundamentalmente de las preferencias de los propios autores.

EJEMPLO 5.32

```
$ x\equiv y\mod n\\
x\equiv y\bmod n\\
x\equiv y\pod n\\
x\equiv y\pmod n $
```

Congruencias: el comando `\mod`

$$\begin{aligned}x &\equiv y \pmod{n} \\x &\equiv y \text{ mód } n \\x &\equiv y \pmod{(n)} \\x &\equiv y \pmod{(mód n)}\end{aligned}$$

5.9. Matemáticas y babel con opción spanish

Hemos comentado en la sección anterior que la presencia de babel con la opción spanish permite utilizar las funciones trigonométricas `\sen`, `\arcsen`, `\tg` y `\arctg` con sintaxis castellanizada. Además de estos valores añadidos, la opción spanish posibilita también:

Funciones acentuadas: Obsérvese en el cuadro 5.16 y en el ejemplo 5.32 que los comandos `\lim`, `\liminf`, `\limsup`, `\max`, `\min`, `\bmod` y `\pmod` proporcionan las versiones acentuadas de acuerdo con las reglas del castellano. Los comandos

`\unaccentedoperators`

`\accentedoperators`

desactivan|activan esta opción en cualquier lugar de nuestro documento. Por defecto se tiene activa la opción que incluye los acentos.

Espacios entre funciones compuestas: Los comandos

`\unspacedoperators`

`\spacedoperators`

desactivan|activan el espacio extra entre «arc» y la función que le sigue. Por defecto se incluye el espacio extra.

i sin punto en modo matemático: La *i* sin punto está disponible en modo matemático mediante el comando `\dotlessi`, de forma que se puede escribir, por ejemplo, `\acute{\dotlessi}`. Al escribir `$V_{\mathbf{cr}\acute{\dotlessi}t}$` se obtiene $V_{\text{crít}}$.

Castellanizar más nombres de funciones: Otras funciones trigonométricas con sintaxis castellana que se encuentran almacenadas en la variable

`\spanishoperators`

son las siguientes:

<code>cotg</code>	<code>\cotg</code>	<code>cosec</code>	<code>\cosec</code>	<code>senh</code>	<code>\senh</code>	<code>tgh</code>	<code>\tgh</code>
-------------------	--------------------	--------------------	---------------------	-------------------	--------------------	------------------	-------------------

Estas funciones se han separado, en su definición, de las que aparecen en el cuadro 5.17, porque, al contrario que `\sen`, `\tg`, la forma de escribirlas está lejos de estar normalizada entre los hispanohablantes. Podemos cambiar la definición de `\cotg`, `\cosec`, etc., incluyendo, por ejemplo, el comando

```
\renewcommand*{\spanishoperators}{ctg arc\,ctg sh ch th}
```

(separadas con espacio). Cuando se selecciona spanish al cargar babel, se crean comandos con los nombres incluidos en la redefinición de `\spanishoperators`. Para esta redefinición, además de las letras sin acentuar se acepta la orden `\`, que se pasa por alto para formar

el nombre. Conviene que `\spanishoperators` esté en el preámbulo del documento, antes de `\selectspanish` si se utiliza éste.

¿Punto o coma en los decimales? En la página 130 se ha explicado que al escribir, por ejemplo, `1.5`, el separador decimal se cambia automáticamente a una coma. Los comandos

`\decimalcomma`

`\decimalpoint`

permiten decidir, en cualquier parte del documento, si se quiere utilizar una coma (valor predeterminado) o un punto, mientras que el comando

`\spanishdecimal{SignoDePuntuación}`

establece el *SignoDePuntuación* que queramos.

5.10. Operadores de tamaño variable: integrales y sumatorios

LOS signos de integración y sumatorio aparecen con profusión en textos matemáticos. Si se lee uno de estos textos se observará que, según que el signo de integración o sumatorio estén en una fórmula que forme parte de un párrafo o esté en una fórmula resaltada, su tamaño es distinto. En L^AT_EX, los signos de integración y sumatorios se obtienen, respectivamente, con los siguientes comandos:

`\int`

`\sum`

En ambos casos, pueden incluirse límites superiores e inferiores. El extremo inferior se escribe como si fuera un subíndice y el superior como si fuera un superíndice.

EJEMPLO 5.33

La igualdad

$$\int_{-\infty}^{+\infty} e^{-x^2} dx = \sqrt{\pi}$$

 es tan clara, para un matemático, como

$$\sum_{n=0}^{\infty} \frac{1}{(2n+1)^2} = \frac{\pi^2}{8}.$$

La suma anterior implica que $\frac{\pi^2}{8} = \sum_{n=1}^{\infty} \frac{1}{(2n+1)^2}$.

Fórmulas, integrales y sumatorios

La igualdad

$$\int_{-\infty}^{+\infty} e^{-x^2} dx = \sqrt{\pi}$$

es tan clara, para un matemático, como

$$\sum_{n=0}^{\infty} \frac{1}{(2n+1)^2} = \frac{\pi^2}{8}.$$

La suma anterior implica que $\frac{\pi^2}{8} = \sum_{n=1}^{\infty} \frac{1}{(2n+1)^2}$.

El tamaño de los símbolos de integral y de sumatorio y la posición de sus límites cambian dependiendo de que la fórmula en la que aparecen esté resaltada en una línea aparte o de que forme parte de un párrafo normal. Este comportamiento de integrales y sumatorios puede modificarse, como en el caso de las fracciones, con las declaraciones `\displaystyle` y `\textstyle` comentadas en la lección 16 y en la sección 5.5.4.

Todos los operadores del cuadro 5.18 tienen un comportamiento similar al que hemos descrito para las integrales y sumatorios.

$\int \backslash int$	$\odot \backslash bigodot$	$\bigoplus \backslash biguplus$	$\prod \backslash prod$
$\oint \backslash oint$	$\oplus \backslash bigoplus$	$\bigvee \backslash bigvee$	$\smallint \backslash smallint$
$\bigcap \backslash bigcap$	$\otimes \backslash bigotimes$	$\wedge \backslash bigwedge$	$\sum \backslash sum$
$\bigcup \backslash bigcup$	$\sqcup \backslash bigsqcup$	$\coprod \backslash coprod^a$	$\coprod \backslash coprod$

^aCuando se tiene cargado el paquete `mathptmx` en su versión actual, `\coprod` no está disponible. La salida \coprod corresponde al carácter '140 de la fuente `cmex` con la codificación «OMX». Se puede declarar un comando para \coprod incluyendo `\DeclareSymbolFont{MiFuente}{OMX}{cmex}{m}{n}` y `\DeclareMathSymbol{\MiCoprod}{\mathop}{MiFuente}{140}` en el preámbulo del documento. Después de estas declaraciones, `\MiCoprod` funciona como un operador de tamaño variable que produce \coprod . Véase la sección PARA SABER MÁS de este capítulo, donde encontrará referencias que le ayudarán a entender las líneas de código anteriores.

Cuadro 5.18: Operadores de acumulación de tamaño variable

5.10.1. Ubicación de los límites en operadores de tamaño variable

Para los operadores de tamaño variable, L^AT_EX decide la posición de los límites, si los hay, en función de la clase de documento que se esté utilizando, y según las fórmulas estén resaltadas o sean parte de un párrafo. A veces, podemos encontrarnos en situación de querer o tener que cambiar la posición de estos límites. Para esto L^AT_EX dispone de los comandos

<code>\limits</code>	<code>\nolimits</code>	<code>\displaylimits</code>
----------------------	------------------------	-----------------------------

cuya finalidad es la siguiente:

`\limits` obliga a que los límites del operador de acumulación de tamaño variable aparezcan centrados bajo los correspondientes signos, independientemente del tipo de fórmula que estemos escribiendo y sin afectar al tamaño del operador;

`\nolimits` es una especie de opuesto a `\limits` que fuerza a que los límites de un operador de acumulación de tamaño variable aparezcan a su derecha;

`\displaylimits` actúa como `\limits` si la fórmula está centrada y como `\nolimits` en caso contrario.

EJEMPLO 5.34

```
\[ \int \limits_{\partial M} f(z) dz = \int \limits_{\partial M} f(z) dz \qquad \sum_{i=1}^n 2^{-i} = \sum_{i=1}^n 2^{-i}
```

$$\int_{\partial M} f(z) dz = \int_{\partial M} f(z) dz \quad \sum_{i=1}^n 2^{-i} = \sum_{i=1}^n 2^{-i}$$

Funcionamiento de los comandos `\limits` y `\nolimits`

5.10.2. Subíndices y superíndices en varias líneas

Para poder escribir varias líneas en los límites de un operador de tamaño variable disponemos del comando

<code>\substack{Líneas}</code>

que admite como argumento las distintas *Líneas* que forman el correspondiente extremo: estas líneas se separan con el comando de salto de línea `\backslash\backslash` o con `\backslash[Salto]`. Las líneas aparecerán centradas dentro del correspondiente límite. Si en lugar de líneas centradas en los límites queremos que éstas aparezcan ajustadas a la izquierda, podemos utilizar el entorno

```
\begin{subarray}{l}
Líneas
\end{subarray}
```

cuyo argumento *Líneas* tiene el mismo significado que para `\substack`. Si se suprime el argumento `l` o se sustituye por `c`, el entorno produce el mismo resultado que el comando `\substack`. Si se cambia `l` por cualquier expresión no vacía, distinta de la letra `c`, las líneas seguirán apareciendo ajustadas a la izquierda.

EJEMPLO 5.35

```
\[\sum_{\begin{subarray}{l}1 \leq i \leq 100 \\ 1 < j < 8\end{subarray}} P(i,j)\]
\[\sum_{\begin{subarray}{l}1 \leq i \leq 100 \\ 1 \leq j < 8\end{subarray}} P(i,j)\]
```

Subíndices con varias líneas

5.10.3. Integrales múltiples

Cuando se quiere utilizar integrales múltiples, la reiteración del comando `\int` puede producir un espacio entre los signos de integración inadecuado. Integrales múltiples con un espacio mejor conseguido son proporcionadas por los comandos

<code>\iint</code>	<code>\iiint</code>	<code>\iiiint</code>	<code>\idotsint</code>
--------------------	---------------------	----------------------	------------------------

El ejemplo que sigue muestra la utilización de estos comandos.

EJEMPLO 5.36

```
\[ \iint \limits_M d\omega
= \iint \limits_{\partial M} \omega \]
\[ \iiint f(x,y,z,w), dx, dy, dz, dw \quad \iiint \limits_M f(x,y,z,w) dx dy dz dw \quad \int \cdots \int_M dx_1 \dots dx_n \]
```

Fórmulas, integrales múltiples

5.10.4. Construyendo operadores de tamaño variable

Como en los casos de símbolos ordinarios, operadores binarios y de relación (véase la sección 5.3.2), puede instruirse a L^AT_EX para que haga de cualquier símbolo o letra existente, o cual-

quier otro que podamos incorporar, un operador de tamaño variable. Para ello disponemos del comando que sigue cuyo comportamiento puede observarse en el ejemplo 5.37.

$\backslash\mathop{Expresión}$

Si se quiere que el operador definido adapte su tamaño según el estilo de la parte de la fórmula en la que interviene (véase la sección 5.5.4), se tiene que definir un nuevo comando usando:

$\backslash\mathchoice{Símbolo-d}{Símbolo-t}{Símbolo-s}{Símbolo-ss}$

donde *Símbolo-?* es cada uno de los símbolos a utilizar según el tipo de fórmula y la situación del operador en la misma: d para *displaystyle*, t para *textstyle*, s para *scriptstyle* y ss para *scriptscriptstyle*.

EJEMPLO 5.37

```
\newcommand*\objeto{\mathop{\mathchoice{%
\textrm{\huge A}}{%
\textrm{A}}{%
\textrm{a}}{%
\textrm{a}}}}\%
\objeto_1^2
\objeto_{\objeto_1^2}
\objeto_{\objeto_{\objeto_1^2}}
```

Ilustración de *mathchoice*

5.11. Diagramas conmutativos con el paquete **amscd**

El paquete **amscd** proporciona el entorno **CD** para escribir diagramas conmutativos rectangulares simples sin flechas en diagonal. Si necesitamos elaborar diagramas más complicados, éstos pueden realizarse mediante paquetes más potentes como **diagram**, **xypic**, **kuvio** y **pstricks** (véase la página 352). El entorno **CD** proporcionado por **amscd** sólo funciona en modo matemático. Su sintaxis es:

```
\begin{CD}
Línea superior \\
Línea central \\
Línea inferior
\end{CD}
```

En las líneas superior e inferior se escriben los nodos del diagrama. Los comandos

@>>	@> <i>Encima</i> > <i>Debajo</i> >
@<<<	@< <i>Encima</i> < <i>Debajo</i> <
@=	

se utilizan para las flechas. Los dos comandos de la fila de arriba producen flechas orientadas hacia la derecha. Los objetos *Encima* y *Debajo* se escriben, respectivamente, sobre la flecha y debajo de la misma. Los dos comandos que les siguen producen flechas orientadas hacia la izquierda. Con @= se obtiene un símbolo de igualdad horizontal con la longitud adecuada al diagrama.

En la línea central se escriben las flechas verticales mediante los comandos:

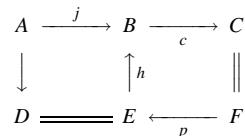
@AAA	$\circ A ObjetoIzquierda A ObjetoDerecha A$
@VVV	$\circ V ObjetoIzquierda V ObjetoDerecha V$
@	

Los dos primeros producen flechas orientadas hacia arriba. Los objetos *ObjetoIzquierda* y *ObjetoDerecha* se escriben, respectivamente, a la izquierda y a la derecha de la flecha. Los dos comandos que les siguen producen flechas orientadas hacia abajo. Con @| se obtiene un símbolo de igualdad en posición vertical con longitud adecuada al diagrama.

El uso del entorno se ilustra en el ejemplo 5.38. Para finalizar llamamos la atención sobre la incompatibilidad del paquete *amscd* y *babel*, cargado con la opción *spanish*, debido a que los símbolos de comillas dobles constituyen un método taquigráfico (que se puede desactivar con el comando *\deactivatequoting*, descrito en la página 176).

EJEMPLO 5.38

```
\deactivatequoting
\[
\begin{CD}
A @>j>> B @>c>> C \\
@VVV @AhA @| \\
D @= E @<p< F
\end{CD}
\activatequoting
```



Diagramas conmutativos con el paquete *amscd*

5.12. Teoremas y demostraciones

HASTA ahora, hemos ido describiendo distintos elementos del lenguaje que se usan para escribir textos matemáticos. En artículos de investigación y en libros de matemáticas, teoremas y demostraciones son elementos comunes. Otros elementos que habitualmente se utilizan para organizar y articular un documento en matemáticas son: definiciones, corolarios, lemas, proposiciones, conjeturas, problemas, etc. De ahora en adelante nos referiremos a estos objetos con el nombre genérico de *entornos tipo teorema*. Con estos entornos se escriben los principales párrafos de un texto matemático. Para poder identificarlos fácilmente deben aparecer resaltados e incluso numerados dentro del texto para, en caso necesario, poder hacer referencias a ellos (véase la lección 8).

Las clases de documento de *LATEX* no proporcionan *entornos* predefinidos de *tipo teorema* porque, de un lado, sería difícil para los autores ejercer el control necesario sobre la numeración de los mismos, y de otro, la variedad de elementos teorema, definición, corolario, etc., es tan grande que sería imposible incluir definiciones para todos. En su lugar *LATEX* dispone del comando *\newtheorem*, similar a *\newenvironment*, que hace posible que un autor pueda definir en su documento tantos *entornos tipo teorema* como necesite, adecuados en nombre, formato y método de numeración a su propio gusto o necesidad.

Un ejemplo típico de utilización de estos entornos es el que sigue. En él definimos dos *entornos tipo teorema* a los que damos los nombres `lem` y `teo`, que posteriormente son invocados usando los comandos `\begin` y `\end` seguidos de los nombres correspondientes. Los argumentos `Lema` y `Teorema` que utilizamos al definir los entornos `lem` y `teo` son, como se observa, los nombres que aparecen impresos al utilizar éstos: los nombres se imprimen en negrita y los *entornos tipo teorema* creados se numeran de forma independiente. Líneas más abajo aprenderemos cómo hacer que la numeración pueda ir ligada.

EJEMPLO 5.39

```
\newtheorem{lem}{Lema}
\newtheorem{teo}{Teorema}
\begin{lem}
Si $\{u_1,u_2,\dots,u_n\}$ es un sistema ...
otros es también un sistema de generadores.
\end{lem}

\begin{teo}
Si $\{v_1,v_2,\dots,v_m\}$ ...
entonces $m \leq n$.
\end{teo}
```

Una primera ilustración de cómo usar `\newtheorem`

Lema 1. Si $\{u_1, u_2, \dots, u_n\}$ es un sistema de generadores de un espacio vectorial V , entonces el conjunto de vectores que se obtiene eliminando los que son combinación lineal de los otros es también un sistema de generadores.

Teorema 1. Si $\{v_1, v_2, \dots, v_m\}$ es linealmente independiente y $\{u_1, u_2, \dots, u_n\}$ es un sistema de generadores de V , entonces $m \leq n$.

La estructura completa de `\newtheorem` es

```
\newtheorem{Tipo} [Contador] {NombreTipo} [ContadorReferencia]
```

donde:

Tipo Es el nombre que damos al nuevo entorno. Al ejecutarse `\newtheorem` se crea el entorno *Tipo* y un contador asociado que lleva el mismo nombre, el cual se incrementa e imprime cada vez que utilizamos `\begin{Tipo}` `\end{Tipo}`.

Contador Es el nombre de otro entorno tipo teorema definido previamente, cuyo contador se utilizará para ligar el contador del entorno que estamos definiendo (véase el ejemplo 5.40).

NombreTipo Es el título que encabezará el entorno al imprimirse éste.

ContadorReferencia Es el nombre del contador de referencia al que el contador de este tipo de entorno está subordinado y que aparecerá junto a éste en su representación (véase el ejemplo 5.40).

Aunque la definición de estos entornos se suele hacer en el preámbulo, se puede realizar en cualquier parte del documento. Conviene tener presente, no obstante, que, en L^AT_EX, el nombre *Tipo* puede utilizarse una única vez en el documento para definir un comando o entorno; debemos ser, por tanto, metódicos y ordenados a la hora de introducir estas nuevas definiciones, para evitar duplicidades.

Los entornos definidos con `\newtheorem{Tipo}` se utilizan con la siguiente estructura:

```
\begin{Tipo} [Comentario]
TextoDelEntorno
\end{Tipo}
```

La utilización de `\begin{Tipo}TextoDelEntorno\end{Tipo}` produce, como hemos visto en el ejemplo 5.39, dos partes diferenciadas: un encabezamiento, que aparece en negrita, formado por *NombreTipo* junto a la numeración correspondiente y a continuación el cuerpo del teorema *TextoDelEntorno*, en itálica. Caso que hayamos hecho uso del argumento opcional *Comentario*, éste se imprimirá a continuación del encabezamiento entre paréntesis y en negrita.

En el ejemplo siguiente subordinamos la numeración de *teo* al contador *section*, lo que hace que al numerarse el primer teorema aparezca el número que le corresponde precedido del contador de la sección actual, que es 5.12. El entorno *cor* aparece ligado al contador *teo*: en la práctica, la numeración de *cor* y *teo* será correlativa. Obsérvese también cómo iniciamos la demostración del corolario y la utilización de los espacios elásticos `\hfill` (véase la sección 1.5.2) para poner la marca de fin de demostración \square elegida por nosotros.

EJEMPLO 5.40

```
\newtheorem*[LZ]{Lema de Zorn}
\newtheorem{teo}{Teorema}[section]
\newtheorem{cor}{Corolario}[teo]

\begin{LZ} Si cada cadena... \end{LZ}

\begin{teo}[Existencia de Bases]
\label{ExistenciaBases}...espacio... \end{teo}

\begin{cor} Toda aplicación...total. \end{cor}
\noindent\emph{Demostración.} Es un sencillo
...utiliza el teorema~\ref{ExistenciaBases}.
\hfill $\square$
```

Cómo usar `\newtheorem` con contadores subordinados

Lema de Zorn. *Si cada cadena en un conjunto parcialmente ordenado tiene cota superior, entonces el conjunto tiene un elemento maximal.*

Teorema 5.12.1 (Existencia de Bases). *Todo espacio vectorial tiene una base de Hamel.*

Corolario 5.12.2. *Toda aplicación lineal que está definida en un subespacio de un espacio vectorial puede extenderse de forma lineal al total.*

Demostración. Es un sencillo ejercicio que utiliza el teorema 5.12.1. \square

En el ejemplo anterior, el entorno `\newtheorem*` (que el propio ejemplo explica) no es parte de L^AT_EX básico y para poder utilizarlo necesitamos cargar el paquete *amsthm* que tiene ésta y otras posibilidades que describimos a continuación.

5.12.1. El paquete *amsthm*

El paquete *amsthm* proporciona una versión mejorada del entorno `\newtheorem` que continúa teniendo la misma estructura que hemos expuesto más arriba. Además de tener el entorno `\newtheorem*`, con *amsthm* se dispone de distintos *estilos* para los teoremas, un entorno *proof* que podemos utilizar para las demostraciones y la posibilidad de elegir la ubicación de la numeración de los teoremas (a la izquierda o derecha de sus etiquetas). El paquete *amsthm* debe cargarse después del paquete *amsmath*.

Hay tres estilos de teoremas disponibles que tipográficamente responden a su importancia en el texto. La declaración de los estilos se hace con el comando

```
\theoremstyle{Estilo}
```

Mediante este comando se establecen los parámetros que utilizará `\newtheorem` en la definición de los entornos tipo teorema a partir de ese momento. En un mismo documento se pueden utilizar distintos estilos. Las declaraciones `\theoremstyle{Estilo}` afectan a los entornos `\newtheorem` definidos a partir de ahí, hasta que una nueva aparición de otra declaración de este tipo sea entonces la que esté vigente. Los *Estilo(s)* reconocidos por `amsthm` son:

`plain` Escribe el encabezamiento del teorema en negrita y el cuerpo en itálica. Estilo predefinido en L^AT_EX.

`definition` Escribe el encabezamiento en negrita y el cuerpo en «roman».

`remark` Escribe el encabezamiento en itálica y el cuerpo en «roman».

En la práctica, para crear entornos tipo teorema con estilos diferentes, es razonable colocar los comandos `\newtheorem` en grupos precedidos por el estilo `\theoremstyle` adecuado. Si no aparece ningún comando `\theoremstyle` en nuestro documento, el estilo que se usa es `plain`; véase el ejemplo siguiente.

EJEMPLO 5.41

```
\newtheorem*[LZ]{Lema de Zorn}
\newtheorem{teo}{Teorema}[section]
\swapnumbers
\newtheorem{cor}[teo]{Corolario}
\swapnumbers
\theoremstyle{definition}
\newtheorem{defi}{Definición}

\begin{LZ} Si cada cadena... \end{LZ}

\begin{teo}[Existencia de Bases]
\label{ExistenciaBases} Todo espacio... \end{teo}

\begin{cor} Toda aplicación..total. \end{cor}
\begin{proof}[Demostración.] Es un... \end{proof}

\begin{defi} Un conjunto de vectores... \end{defi}
```

Utilidades de `amsthm`

Lema de Zorn. *Si cada cadena en un conjunto parcialmente ordenado tiene cota superior, entonces el conjunto tiene un elemento maximal.*

Teorema 5.12.3 (Existencia de Bases). *Todo espacio vectorial tiene una base de Hamel.*

5.12.4 Corolario. *Toda aplicación lineal definida en un subespacio de un espacio vectorial puede extenderse de forma lineal al total.*

Demostración. Es un sencillo ejercicio que utiliza el teorema 5.12.3. □

Definición 1. Un conjunto de vectores en \mathbb{R}^n se dice...

Es posible cambiar la posición entre el nombre del entorno teorema y su número. Esto se realiza mediante el comando

`\swapnumbers`

situado delante de la declaración `\newtheorem` del entorno tipo teorema para el que queremos que se intercambien número y nombre. Este comando es independiente del estilo elegido y ha sido diseñado para poder repetirlo varias veces. Cada vez que se utiliza, sitúa los números en el lado opuesto de lo que se venía haciendo hasta ese momento; véanse en el ejemplo anterior las líneas anterior y posterior a la definición del entorno `cor`.

Como dijimos anteriormente, `amsthm` también tiene definido el entorno `proof` para escribir demostraciones. Si escribimos

```
\begin{proof}[EncabezamientoOpcional]
Prueba
\end{proof}
```

se produce un encabezamiento «*Proof.*» y se añade el símbolo \square al final de la última línea de la *Prueba*. Como argumento *EncabezamientoOpcional* se puede poner un encabezamiento en sustitución de «*Proof.*». Y si no nos gusta el símbolo \square o su posición, podemos redefinir el comando `\qed` que es el que *proof* utiliza para poner la marca de fin de demostración. Si un entorno `\begin{proof} \end{proof}` acaba en una fórmula resaltada, el símbolo de final de demostración pasa a la siguiente línea, lo que es estéticamente lamentable. Este problema se corrige con el comando `\qedhere`, como muestra el ejemplo siguiente. Si `\qedhere` produce un error, puede tratar de subsanarse incluyendo `\boxed{\qedhere}`.

EJEMPLO 5.42

```
\renewcommand*\proofname{Demostración}
\begin{proof}...$1+1=2...$1+2=3.\end{proof}

\begin{proof}...$1+1=2$...[1+2=3.] \end{proof}

\begin{proof}...$1+1=2$ se obtiene que \[1+2=3.
\qedhere\] \end{proof}

\renewcommand*\qed{\hfill$\blacksquare$}
\begin{proof}...$1+1=2$...$1+2=3.\end{proof}
```

Demostración. Teniendo en cuenta que $1 + 1 = 2$ se obtiene que $1 + 2 = 3$. \square

Demostración. Teniendo en cuenta que $1 + 1 = 2$ se obtiene que

$$1 + 2 = 3.$$

\square

Demostración. Teniendo en cuenta que $1 + 1 = 2$ se obtiene que

$$1 + 2 = 3.$$

\square

Demostración. Teniendo en cuenta que $1 + 1 = 2$ se obtiene que $1 + 2 = 3$. \blacksquare

Diversas marcas de fin de demostración

La variable `\proofname` guarda el encabezamiento con el que empiezan las demostraciones. Su redefinición mediante `\renewcommand*\{\proofname\}{Demostración}` traduce al castellano el encabezamiento *Proof* del entorno *proof*.

5.13. Parámetros globales en las fórmulas matemáticas

En esta sección relacionamos las longitudes que se utilizan en modo matemático. Los valores que tienen asignados por defecto dependen de la clase de documento y del tamaño de las fuentes que se estén utilizando.

`\abovedisplayskip` `\belowdisplayskip` Controlan la separación vertical dejada antes (después) de una fórmula resaltada cuando ésta empieza en un punto más cercano al margen izquierdo que el punto donde acabó la línea anterior. Con la opción `fleqn` se utilizará, sin embargo, `\topsep`.

`\abovedisplayshortskip` `\belowdisplayshortskip` Controlan la separación vertical dejada antes (después) de una fórmula resaltada cuando ésta empieza en un punto más alejado

del margen izquierdo que el punto donde acabó la línea anterior. Con la opción `fleqn` se utilizará, sin embargo, `\topsep`.

`\mathindent` Si la opción `fleqn` está seleccionada, indica el sangrado de las fórmulas matemáticas en estilo `display` (excepto las obtenidas mediante `$$...$$`).

`\jot` Es la separación vertical extra entre dos ecuaciones consecutivas de un entorno `eqnarray` o `eqnarray*`.

En el apartado 3.3.1 se han descrito los parámetros de las tablas que coinciden con los correspondientes a las matrices, y que por tanto no repetimos aquí.

En el ejemplo que sigue utilizamos las longitudes anteriores para modificar la amplitud de los espacios antes y después de fórmulas resaltadas.

EJEMPLO 5.43

```
Por ejemplo, la fórmula \abovedisplayskip.5cm
\belowdisplayskip .5cm
\[e^{i\pi}+1=0\]
se ha obtenido con \verb+\abovedisplayskip.5cm+
y \verb+\belowdisplayskip.5cm+, cuando lo normal
hubiera sido lo siguiente \abovedisplayskip10pt
\belowdisplayskip10pt
\[e^{i\pi}+1=0\]
```

Ej.: \abovedisplayshortskip.5cm
\belowdisplayshortskip.5cm
\[\cos^2(\theta)+\sin^2(\theta)=1\]
lo escribe el \LaTeX{} cuando\dots

```
\jot1pt\begin{eqnarray*}
x + y & = & z \\
a + b & = & c \\
1 + 2 & = & 3\end{eqnarray*}

\jot10pt\begin{eqnarray*}
x + y & = & z \\
a + b & = & c \\
1 + 2 & = & 3\end{eqnarray*}
```

Distintas longitudes en modo matemático

Por ejemplo, la fórmula

$$e^{i\pi} + 1 = 0$$

se ha obtenido con \abovedisplayskip .5cm y \belowdisplayskip .5cm, cuando lo normal hubiera sido lo siguiente

$$e^{i\pi} + 1 = 0$$

Ej.:

$$\cos^2(\theta) + \sin^2(\theta) = 1$$

lo escribe el \LaTeX{} cuando...

$$\begin{array}{rcl} x+y & = & z \\ a+b & = & c \\ 1+2 & = & 3 \end{array}$$

$$\begin{array}{rcl} x+y & = & z \\ a+b & = & c \\ 1+2 & = & 3 \end{array}$$

PARA SABER MÁS

- La página web de la *American Mathematical Society* (A.M.S.) (en <http://www.ams.org>) tiene un apartado con el nombre *Reference tools | Tex Resources*, en el que encontrará numerosas herramientas que le pueden ser de utilidad, si tiene que escribir documentos matemáticos. En particular, encontrará los productos *AMS-LaTeX*, *AMS-TeXy* *AMS-Fonts*, parte de los cuales ya tendrá usted instalados en su sistema, puesto que alguno de ellos se

distribuye con la versión estándar de L^AT_EX. Quizás sea también de su interés saber que, formando parte de estas distribuciones, se encuentran las clases `amsart` y `amsbook`, que han sido diseñadas para escribir, respectivamente, artículos y libros publicados en un estilo similar a las publicaciones de la A.M.S. También encontrará en la página web mencionada el novísimo paquete `amsref...`; este paquete permite controlar las listas de bibliografía a través de L^AT_EX, pudiendo obviar la utilización de BIBL^AT_EX; L^AT_EX lee directamente de una base de datos que contiene los datos en la misma estructura que aparecen en el documento.

- En los pies de los cuadros 5.5 y 5.18, y del ejemplo 5.4, han sido utilizados los comandos `\DeclareSymbolFont` y `\DeclareMathSymbol`, para declarar nuevos símbolos matemáticos a partir de un determinado carácter de una fuente. En la sección 7.7.6 de [21] encontrará explicaciones sobre los comandos anteriores, y en general, explicaciones de cómo declarar nuevas fuentes para utilizarlas en modo matemático.

LA navegación a través de Internet consiste esencialmente en ir abriendo documentos que contienen enlaces de hipertexto, a través de los cuales «viajamos» de un documento a otro. La mayor parte de las páginas que visitamos, y de los documentos que abrimos, están en formato HTML o PDF. Este capítulo tiene por objetivo mostrar cómo construir con L^AT_EX documentos en estos dos formatos, adecuados para ser publicados en Internet. Para ello completaremos la información para construir documentos PDF que hemos ido aportando desde la primera lección y presentaremos una herramienta capaz de generar documentos HTML a partir de nuestros documentos L^AT_EX.

La primera sección la dedicamos a completar el análisis del paquete hyperref que iniciamos en las lecciones 8 y 15. Como ya sabemos, con sólo cargar el paquete las referencias cruzadas se convierten en enlaces de hipertexto; aprenderemos ahora a crear enlaces más generales.

En la segunda sección ampliamos el abanico de posibilidades de los documentos PDF indicando cómo incluir en ellos anotaciones de diferentes tipos: notas de texto, enlaces que ejecutan acciones, sonidos y películas.

Por último, en la tercera sección presentaremos el sistema de conversión T_EX4ht para convertir documentos L^AT_EX en documentos HTML. El compilador L^AT_EX juega un papel principal en estas conversiones pues se encarga de realizar la primera traducción al compilar el documento fuente con el paquete tex4ht. Describiremos las distintas opciones de conversión y algunos comandos que proporciona el paquete para controlar el estilo de las páginas HTML que se obtienen.

6.1. Hiperenlaces en L^AT_EX con el paquete hyperref

ESTE paquete, desarrollado por S. Rathz y H. Oberdiek [57], convierte automáticamente en enlaces de hipertexto todos los tipos de referencias que L^AT_EX establece. En las lecciones 8 y 15 de la primera parte ya hemos analizado cómo trata las referencias cruzadas usuales y las citas bibliográficas. También hemos aprendido a realizar manualmente enlaces de hipertexto, y algunos de los comandos básicos que el paquete pone a nuestra disposición para enlaces con documentos externos. Uno de los comandos básicos para crear enlaces de hipertexto es

```
\href{URL}{Objeto}
```

dvipdf	dvipdfm	dvips	dvipsone	dviwindo
hypertex	latex2html	nativepdf	pdfmark	pdftex
ps2pdf	tex4ht	textures	vtex	vtexpdfmark

Cuadro 6.1: Controladores admitidos por el paquete hyperref. Las opciones nativepdf y pdfmark son equivalentes a dvips

que convierte el argumento *Objeto* en un hiperenlace que conecta con el destino *URL*. En general, un destino *URL* tiene dos partes, separadas por dos puntos:

Tipo: Dirección_Completa

En la lección 8 vimos un tipo especial de URL construido con la partícula `file`, utilizado para crear enlaces con otros documentos PDF. Algunos de los posibles tipos de URL son los siguientes:

http Es el tipo por defecto y no es necesario indicarlo. En este caso, tampoco es necesario incluir ni los dos puntos ni la doble barra que suele aparecer en las direcciones de Internet. Así, por ejemplo, las dos líneas siguientes son equivalentes:

```
\href{http://www.um.es/}{Conectar con la Universidad de Murcia}
\href{www.um.es}{Conectar con la Universidad de Murcia}
```

ftp Permite conectar con servidores FTP («File Transfer Protocol») utilizados para descargas de archivos por Internet. En este caso es necesario incluir tanto los dos puntos como la doble barra.

```
\href{ftp://ftp.um.es/}{Servidor de la Universidad de Murcia}
```

mailto Para enviar un correo electrónico, utilizando el programa que nuestro explorador de Internet tenga predefinido.

```
\href{mailto:latex@um.es}{Enviar un correo a los autores}
```

file Abre un archivo con el programa asociado. Puede que sea necesario incluir también el directorio donde se encuentra el archivo.

```
\href{file:c:/windows/control.ini}{Abrir el archivo control.ini}
```

run Similar al anterior, permite ejecutar una aplicación.

```
\href{run:c:/windows/notepad.exe}{Ejecutar el Bloc de Notas}
```

Ya se indicó en la lección 8 que los dos principales «controladores» del paquete eran `dvips` y `pdftex`, diseñados para crear archivos PS y PDF, respectivamente. Pero el paquete soporta muchos más, que se recogen en el cuadro 6.1.

El paquete `hyperref` redefine muchos comandos de L^AT_EX, por lo que sus autores recomiendan que sea el último en ser cargado, para garantizar así su correcto funcionamiento. No obstante, incluso cargándolo en último lugar, pueden producirse errores, debidos a la incompatibilidad con otros paquetes cargados anteriormente; lea el apartado 6.1.7 para saber si el paquete funciona correctamente en su computador.

Los diferentes parámetros y opciones que el paquete soporta se pueden indicar en el momento de cargar el paquete, siguiendo el esquema «keyval» (v. pág. 74), pues el paquete carga automáti-

camente el paquete keyval. Otra posibilidad, que recomendamos fuertemente, es la de especificar un valor para los parámetros y opciones utilizando el comando \hypersetup, que ya fue introducido en la primera parte, concretamente en la página 75.

No obstante, cuando se carga el paquete, éste lee, si existe, el archivo hyperref.cfg, que puede contener el valor por defecto de algunas opciones (entre ellas el controlador que se utilizará). Por ejemplo, un archivo hyperref.cfg típico podría contener las siguientes líneas:

```
\@ifundefined{pdfoutput}{\ExecuteOptions{dvips}}{\ExecuteOptions{pdftex}}
\hypersetup{colorlinks, pdfstartview=Fit, pdfpagemode=UseThumbs}
```

En resumen, hay cuatro maneras de configurar el comportamiento del paquete, que actúan siguiendo el siguiente orden:

1. con el contenido del archivo hyperref.cfg;

2. con las opciones globales de la clase de documento, que se transfieren al paquete:

```
\documentclass[pdftex]{article}
```

3. con las opciones locales del paquete:

```
\usepackage[pdftitle={Jugar con hyperref}, colorlinks=false]{hyperref}
```

4. con los diferentes comandos \hypersetup, que pueden aparecer en el preámbulo o en el cuerpo del documento.

A continuación listamos las opciones del paquete agrupadas por su funcionalidad. Entre paréntesis figura el valor por defecto, cuando exista para la referida opción.

6.1.1. Configuración de los enlaces

Todas las opciones de este apartado son variables lógicas, es decir, sólo admiten los valores true (verdadero) o false (falso). Debemos recordar que si se les quiere asignar el valor true bastará incluir el nombre de la opción, sin necesidad de la parte =true.

raiselinks (false). La altura de los enlaces con el controlador hypertex es calculada normalmente como la altura de una interlínea de texto. Esta opción obliga a calcular la altura real del enlace, que eventualmente podría contener un objeto de amplias dimensiones (como un gráfico, por ejemplo).

breaklinks (false). Permite que los enlaces puedan ocupar varias líneas. Como esta cualidad no se acomoda bien a las normas de construcción de los archivos PDF, esta opción sólo es activada por defecto con el controlador pdftex. Consecuentemente, los enlaces se dividen en varios enlaces dirigidos todos al mismo punto de destino.

frenchlinks (false). Utiliza letras versalitas para la construcción de los hiperenlaces, en lugar de utilizar colores.

pageanchor (true). Determina si todas las páginas van a contener una marca implícita en la esquina superior izquierda, que se utilizará como punto de destino para hiperenlaces. Si esta opción está desactivada, entonces las entradas del índice general no se convertirán en enlaces a las correspondientes páginas.

`plainpages (true)`. Obliga a que las marcas de las páginas proporcionadas por la opción `pageanchor` se nombren según la forma arábiga del contador de las páginas, en lugar de utilizar su representación. En otras palabras, si esta opción está activada, entonces la etiqueta de la marca de las páginas será `\arabic{page}` y, en caso contrario, la etiqueta será `\thepage` que, eventualmente, puede tener una definición diferente.

6.1.2. Configuración de los colores

Como sabemos, el paquete permite la utilización de colores para la construcción de los hiperenlaces. Existen dos maneras diferentes de colorear los hiperenlaces: coloreando el texto (más generalmente, el objeto) que determina el hiperenlace, o recuadrando de un determinado color dicho objeto.

Según la opción escogida, la determinación de los colores se realiza de un modo diferente. Todo depende de la opción `colorlinks` (v. lección 8), que activa los colores en los hiperenlaces. Si `colorlinks` está activada, entonces disponemos de las opciones descritas a continuación:

`linkcolor (red)`. Color para los enlaces normales.

`anchorcolor (black)`. Color para las marcas que se incluyen en las páginas del documento si la opción `pageanchor` está activada.

`citecolor (green)`. Color para las citas bibliográficas.

`filecolor (cyan)`. Color para los enlaces con archivos locales.

`menucolor (red)`. Color para los enlaces a las entradas de menú del programa ACROBAT.

`pagecolor (red)`. Color para las referencias a otras páginas del documento.

`urlcolor (magenta)`. Color para los enlaces con direcciones URL de Internet.

Los colores utilizados deben ser colores definidos en L^AT_EX a través del paquete `color` siguiendo el modelo `named` (véanse la lección 5 y la sección 3.1).

Por el contrario, si `colorlinks` no está activada, entonces disponemos de las siguientes opciones para especificar el color de los recuadros que se construirán alrededor de los hiperenlaces. Conviene destacar que ahora los colores de los bordes de las cajas que contienen los enlaces deben ser especificados con tres números entre 0 y 1 (según las especificaciones RGB, véase la pág. 274) y no utilizando nombres de colores, como ocurría en el caso anterior.

`pdfborder (0 0 1)`. Especifica el estilo del recuadro que rodea los enlaces: son tres números que indican, respectivamente, los radios horizontal y vertical en las esquinas (que son arcos de elipse) y el grosor de la línea, medidos en pt.

`linkbordercolor (1 0 0)`. Color del recuadro que rodea los enlaces ordinarios.

`citebordercolor (0 1 0)`. Color del recuadro de las citas bibliográficas.

`filebordercolor (0 .5 .5)`. Color del recuadro que rodea los enlaces a otros archivos locales.

`menubordercolor (1 0 0)`. Color del recuadro de los enlaces al menú del programa ACROBAT.

`pagebordercolor (1 1 0)`. Color del recuadro que rodea los enlaces a las páginas.

`urlbordercolor (0 1 1)`. Color del recuadro que rodea los enlaces a las direcciones URL de Internet.

6.1.3. Opciones de visualización específicas de PDF

El programa ACROBAT permite desplazarse a posiciones específicas en los documentos PDF mediante el uso de «marcadores», «miniaturas», «hiperenlaces» (también llamados «vínculos») y «artículos» (una especie de hilos electrónicos que guían al usuario por un documento). A continuación se listan las opciones que permiten especificar el modo en que se visualizará nuestro documento PDF. Conviene señalar que algunas de ellas sólo pueden ser utilizadas en el preámbulo del documento, lo que significa que si se utilizan después no tienen ningún efecto; estas opciones están señaladas con el símbolo \textcircled{P} .

`bookmarks` (true). Por defecto se genera un archivo de extensión `out` que tiene una estructura similar al índice general, pero cuyas entradas se utilizan como marcadores en el archivo PDF. Sin embargo, en ciertas ocasiones es posible que hayamos modificado manualmente este archivo, por lo que es conveniente que, en sucesivas compilaciones, L^AT_EX no lo vuelva a sobreescribir. Para ello deberemos desactivar esta opción, de modo que el archivo de extensión `out` no se generará.

`bookmarksnumbered` (false). Especifica si los números de las secciones deben ser incluidos en los diferentes ítems de la página de marcadores.

`bookmarksopen` (false). Cuando esta opción está activada ACROBAT muestra todos los marcadores completamente expandidos.

`bookmarksopenlevel` (`\maxdimen`). Nivel máximo hasta el que serán mostrados los marcadores.

`pdffagemode` \textcircled{P} Determina el modo en que se abrirá el archivo con el programa ACROBAT. Los posibles valores de esta opción son:

- `None`: no visualiza ni los marcadores ni las miniaturas;
- `UseThumbs`: visualiza las miniaturas;
- `UseOutlines`: visualiza los marcadores;
- `FullScreen`: visualiza el documento a pantalla completa.

Si no se selecciona un modo pero se ha activado la opción `bookmarks`, entonces toma el valor `UseOutlines` (véase la figura 6.1).

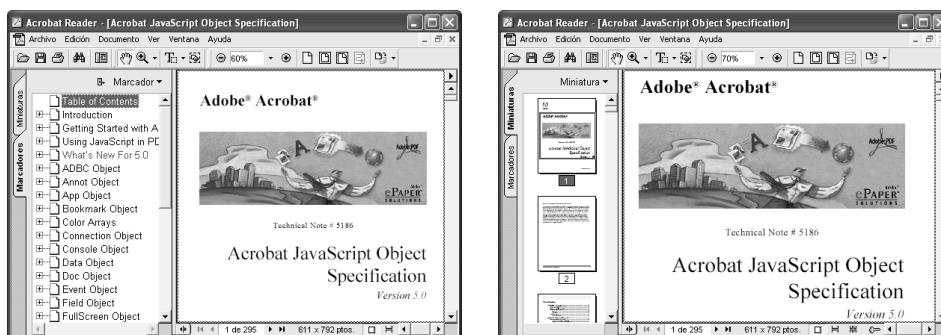


Figura 6.1: Diferentes formas en las que se puede visualizar un documento PDF: a la izquierda mostrando los marcadores y a la derecha las miniaturas

pdfpagelayout (P) Especifica la forma en la que se visualizarán las diferentes páginas del documento. Los posibles valores de esta opción son:

- **SinglePage**: visualiza las páginas de una en una;
- **OneColumn**: visualiza las páginas en una columna;
- **TwoColumnLeft**: visualiza las páginas en dos columnas, situando a la izquierda las páginas impares;
- **TwoColumnRight**: visualiza las páginas en dos columnas, situando a la derecha las páginas impares.

pdftoolbar (true). (P) Determina si ACROBAT mostrará la «barra de herramientas».

pdfmenubar (true). (P) Determina si ACROBAT mostrará la «barra de menús».

pdffitwindow (false). (P) Determina si ACROBAT debe adaptar su tamaño al de la primera página visualizada del documento.

pdfcenterwindow (false). (P) Determina si ACROBAT debe situarse en el centro del monitor.

pdfwindowui (P) Especifica si los elementos de la ventana interfaz de usuario (por ejemplo, la ventana de marcadores) deben estar visibles (valor **true**) u ocultos (valor **false**).

pdfstartpage (1). (P) Indica la página que se visualizará en primer lugar al abrir el documento PDF.

pdfstartview (FitB). (P) Determina el modo en que se visualizará el documento, y sus posibles valores quedan recogidos en el cuadro 6.2. Algunos ejemplos son los siguientes:

```
\hypersetup{pdfstartview=FitBH 100}
\hypersetup{pdfstartview=FitBV 200}
\hypersetup{pdfstartview=XYZ 200 300 4}
```

Ajuste	Descripción
Fit	Ajusta la página completa a la ventana.
FitB	Ajusta el texto a la ventana.
FitH Sup	Ajusta la anchura de la página a la ventana.
FitBH Sup	Ajusta la anchura del texto a la ventana.
FitV Izq	Ajusta la altura de la página a la ventana.
FitBV Izq	Ajusta la altura del texto a la ventana.
XYZ Izq Sup Zoom	El parámetro <i>Zoom</i> indica el factor de escala (el valor 1 equivale al 100 %)

Cuadro 6.2: Modos de visualización en un documento PDF. Los valores *Izq* y *Sup* son opcionales (salvo en el modo *XYZ*) y especifican la distancia desde el origen de la página a la parte izquierda o superior, respectivamente, de la ventana, expresado en unidades bp

pdfpagetransition Determina cómo debe realizarse la transición entre las páginas del documento PDF. Los posibles valores se indican en el cuadro 6.3. Algunos ejemplos de transiciones son los siguientes:

```
\hypersetup{pdfpagetransition={Split /M /O}}
```

```
\hypersetup{pdfpagetransition={Glitter /Di 180 /D 2}}
```

pdfpagescrop (P) Esta opción requiere cuatro números que permiten especificar la región máxima de las páginas que será visualizada. Su sintaxis es:

```
pdfpagescrop={a b c d}
```

donde a indica el margen izquierdo, b es el margen inferior, c es el margen derecho y d es el margen superior, todo ello expresado en unidades bp. Esto significa que la anchura de la nueva página será (c-a)bp y la altura (d-b)bp. Por ejemplo, si escribimos

```
pdfpagescrop={100 200 400 700}
```

entonces vamos a visualizar la región obtenida al recortar 100 bp por la izquierda, 200 bp por la parte inferior y que tiene unas dimensiones de 300 bp × 500 bp.

pdfhighlight (/l). Indica cómo se comportan los hiperenlaces cuando son seleccionados. Los valores posibles son: /l (se muestra en colores inversos), /N (sin efecto), /O (se visualiza un cuadro), /P (se comporta como un botón).

6.1.4. El resumen de un documento PDF

Cuando visualizamos un documento PDF en ACROBAT, al seleccionar el elemento de menú Archivo|Propiedades del documento|Resumen tenemos acceso a una pantalla de información donde aparecen diversos datos referentes al documento. Para configurar dicha información, el paquete hyperref dispone de las siguientes opciones.

pdftitle Información que aparece en el campo «Título».

pdfauthor Información que aparece en el campo «Autor».

pdfsubject Información que aparece en el campo «Asunto».

pdfkeywords Información que aparece en el campo «Palabras clave».

pdfcreator Información que aparece en el campo «Creador». Por defecto, su valor es ‘LaTeX with hyperref package’.

pdfproducer Información que aparece en el campo «Producido en».

Una precaución importante que debemos tomar es que el texto que incluyamos en las diferentes opciones tiene que seguir las especificaciones del lenguaje PDF. Como esto no es fácil de recordar, ni de utilizar, el paquete hyperref pone a nuestra disposición el siguiente comando:

```
\pdfstringdef{\Textopdf}{Text}
```

que almacena *Texto* en la variable \Textopdf siguiendo las normas de codificación del lenguaje PDF. De esta forma podemos utilizar un comando auxiliar para ir introduciendo los diferentes campos, como se indica a continuación:

```
\pdfstringdef{\Infor}{¿Qué es un positrón?}\hypersetup{pdftitle=\Infor}
\pdfstringdef{\Autor}{Tomás Nuñez}\hypersetup{pdfauthor=\Autor}
```

y así sucesivamente con el resto de los campos de información. Otro comando que también nos puede ayudar a introducir texto siguiendo la codificación PDF, aunque no es ésta su principal finalidad, es el siguiente:

Efecto	Parámetros	Descripción
Split	/Dm, /M	Dos líneas se separan (horizontal o verticalmente, según la clave /Dm) a lo largo de la pantalla mostrando la nueva página, de forma similar a la apertura de una cortina. El movimiento comienza en el centro o en la orilla, según el parámetro /M.
Blinds	/Dm	Similar a Split, pero utilizando varias líneas, como en una cortina veneciana. El parámetro /Dm determina si el movimiento es horizontal o vertical.
Box	/M	Una caja crece desde el centro de la página vieja, o decrece hacia el centro según la clave /M, para mostrar la nueva.
Wipe	/Di	Una sola línea cruza la pantalla mostrando la nueva página, según la dirección determinada por /Di.
Dissolve		La página anterior se disuelve mostrando la nueva.
Glitter	/Di	Similar a Dissolve, excepto que el efecto se traslada de una parte a la opuesta, según la dirección determinada por /Di.
R		La página anterior es simplemente reemplazada por la nueva, sin ningún efecto especial.

Parámetro	Tipo	Descripción
/D	real	Duración de la transición en segundos (aplicable a todos los efectos).
/Di	real	Dirección del movimiento. Sólo se permiten múltiplos de 90°.
/Dm	nombre	Posibles valores son /H o /V para los efectos horizontal y vertical, respectivamente.
/M	nombre	Especifica si el efecto se inicia en el centro o en el exterior. Los valores posibles son /I para el interior y /O para el exterior.

Cuadro 6.3: Efectos de transición entre páginas de un documento PDF y parámetros de los efectos

```
\texorpdfstring{Objeto_TeX}{Objeto_PDF}
```

Este comando permite incluir la misma información usando dos codificaciones distintas. Al compilar el documento con PDFLATEX, si el destino del comando es el texto del documento PDF entonces se utiliza *Objeto_TeX*, mientras que si el destino es el propio documento PDF entonces se utiliza *Objeto_PDF*, siguiendo la codificación del lenguaje PDF. De este modo, la información podría incluirse como sigue:

```
\hypersetup{pdfsubject=\texorpdfstring{}{}%
           {Interpretación del positrón y su antipartícula}}
```

6.1.5. Acceso a los menús de ACROBAT

El paquete proporciona un comando específico para acceder a los menús de las aplicaciones ACROBAT, disponible sólo en los controladores que producen como salida un documento PDF. El nombre y la sintaxis del comando es la siguiente:

```
\Acrobatmenu{MenúItem}{Objeto}
```

El *Objeto* es utilizado para crear un botón que activa la apropiada opción *MenúItem* del programa. Las principales opciones disponibles como argumento *MenúItem* se listan en el cuadro 6.4, aunque debemos señalar que la disponibilidad de las mismas depende de la versión que estemos utilizando, así como si nos referimos a ACROBAT READER o a la versión completa de ACROBAT que permite la edición y modificación de los documentos PDF. Algunos ejemplos son los siguientes:

```
\Acrobatmenu{GeneralPrefs}{Preferencias del programa}
```

```
\Acrobatmenu{GeneralInfo}{\colorbox{Blue}{\color{White}Resumen}}
```

6.1.6. Opciones adicionales

Algunas otras opciones del paquete, con objetivos muy diversos, son las siguientes:

`linktocpage` (false). En los índices del documento (general, de cuadros y de figuras) suele ser el texto el que se transforma en hiperenlace. Con esta opción activada, se convierten en hiperenlaces sólo los números de las páginas.

`baseurl` ⓘ Asigna la dirección URL base del documento PDF.

`extension` Asigna la extensión de los archivos (por ejemplo, pdf o dvi) que será añadida en los enlaces a otros archivos, siempre que se utilice el paquete `xr` (véase el apartado PARA SABER MÁS del capítulo 2).

`hyperfigures` (false). Convierte los comandos de inclusión de gráficos en hiperenlaces de hipertexto a direcciones URL de Internet que contienen los archivos gráficos.

`backref` (false). Añade, al final de cada entrada de la bibliografía, una lista con los números de las secciones en las que se cita. Esta opción sólo funciona correctamente si después de cada entrada `\bibitem` existe una línea en blanco.

`pagebackref` (false). Añade, al final de cada entrada de la bibliografía, una lista con los números de las páginas en las que se cita.

`hyperindex` (false). Convierte las entradas de los índices terminológicos en enlaces de hipertexto.

6.1.7. ¿Problemas con hyperref?

El paquete redefine muchos comandos de L^AT_EX y de algunos de los paquetes más usuales, lo cual puede originar serios problemas si, con posterioridad a la versión del paquete `hyperref`, se modifican estos comandos. Para verificar que ninguno de estos cambios haya afectado al funcionamiento del paquete, se proporciona un fichero de prueba, cuyo nombre es `hycheck.tex` y cuya finalidad es comprobar si el paquete `hyperref` funcionará correctamente en nuestro documento. Al componer este documento, en pantalla irán apareciendo líneas como las siguientes:

Entrada principal	Opciones disponibles
File	Open, Web2PDF:OpnURL, OpenAsPDF, Close, Save, SaveAs, Revert, AcroSendMail:SendMail, Exit
File Import	Scan, ImportNotes, AcroForm:ImportFDF
File Export	ExtractImages:JPEG, ExtractImages:PNG, ExtractImages:TIFF, ExportNotes, AcroForm:ExportFDF
File Document Properties	GeneralInfo, OpenInfo, FontsInfo, Weblink:Base
Edit	Undo, Redo, Cut, Copy, Paste, Clear, CopyFileToClipboard, SelectAll, DeselectAll, Find, FindAgain, Properties
Edit Preferences	GeneralPrefs, DocBox:Prefs, Web2PDF:InetControlPanel, Web2PDF:Prefs, BCLC:Table#2FFormatted_TextPreferences
Document	FirstPage, PrevPage, NextPage, LastPage, GoToPage, GoBackDoc, GoBack, GoForward, GoForwardDoc, InsertPages, ExtractPages, ReplacePages, DeletePages, CropPages, RotatePages, NumberPages, AcroForm:Actions
View	FullScreen, ZoomViewIn, ZoomViewOut, ZoomTo, FitPage, ActualSize, FitWidth, FitVisible, Reflow, SinglePage, OneColumn, TwoColumns, RotateCW, RotateCCW, ProofColors, Overprint, UseLocalFonts, ShowGrid, SnapToGrid
View Proof Setup	ProofCustom, ProofInkBlack, ProofPaperWhite
Window	Cascade, CloseAll, ShowHideMenuBar, ShowHideClipboard, ShowHideArticles, ShowHideBookmarks, ShowHideAnnotManager, ShowHideDestinations, ShowHideFields, ShowHideInfo, ShowHideSignatures, ShowHideTags, ShowHideThumbnails
Window Tile	TileHorizontal, TileVertical
Window Toolbars	ShowHideAdobe#20&Online, ShowHide&Basic#20Tools, ShowHide&Commenting, ShowHide&Editing, ShowHide&File, ShowHide&Navigation, ShowHideView#20&History, ShowHide&Viewing, ShowHideToolBar
Help	HelpUserGuide, TopIssues, AdobeOnline, Registration, AcroForm:FormsJSGuide, About, AboutAdobeExtensions, BCLC:Table#2FFormatted_TextAbout, DocBox:About

Cuadro 6.4: Opciones de menú del programa ACROBAT

- * Format: ‘LaTeXe’
- Loaded: ‘LaTeXe’ 2001/06/01
- * Checking ‘\addcontentsline’:
- ‘\addcontentsline’ ok.

Si todos los «Checking» son «ok», entonces el paquete funcionará perfectamente.

6.2. Incluyendo anotaciones en los documentos PDF con PDFLATEX

AS anotaciones en un documento PDF son notas u otros objetos que están asociados con una página pero que se encuentran perfectamente diferenciados de la descripción de la página. Los documentos PDF soportan varias clases de anotaciones, como son las «notas de texto» (para consultas interactivas: dicho texto no se visualiza ni aparece en la impresión, aunque se puede

acceder a él a través de un símbolo gráfico que se visualiza en el lugar de la anotación), los «enlaces de hipertexto» que ya hemos analizado en la sección 8.3, los «sonidos» y las «películas», aunque estos dos últimos podrían considerarse de un mismo tipo común «multimedia». Todas las anotaciones pueden manejarse de forma muy sencilla en PDFTEX; para la inserción de las anotaciones se dispone de los dos comandos siguientes:

```
\pdfannot width Anchura height Altura depth Profundidad {ComandosYContenido}  
\pdflastannot
```

El primero permite construir una anotación con las dimensiones dadas por los parámetros *Anchura*, *Altura* y *Profundidad*, mientras que el segundo identifica la última anotación creada con \pdfannot. Más adelante volveremos a insistir en el primer comando, que ilustraremos con numerosos ejemplos, pero ahora es el momento de que detallemos los diferentes elementos y características de cada uno de los distintos tipos de anotaciones.

Los atributos comunes a todas las anotaciones se recogen en el cuadro 6.5, donde los diferentes tipos de atributos que aparecen tienen la siguiente interpretación:

nombre es una palabra válida en lenguaje PDF.

matriz es un conjunto de números (generalmente tres o cuatro) separados entre sí por un espacio blanco y colocados entre corchetes, por ejemplo, [12 20 450 678 1 0].

rectángulo es una matriz de cuatro números, por ejemplo, [0 5 15 40].

fecha es una sucesión de números con formato AAAAMMDDHHMMSS, que indican año, mes, día, hora, minutos y segundos, y que se suele escribir como (D:AAAAMMDDHHMMSS).

cadena es una sucesión de palabras válidas en lenguaje PDF entre paréntesis, por ejemplo, (Esto es una cadena).

A continuación vamos a describir las diferentes anotaciones, indicando, para cada una, cómo se pueden incluir en nuestro documento PDF utilizando PDFTEX.

6.2.1. Notas de texto

Una anotación de texto contiene una cadena de texto que se visualiza cuando la anotación es activada. La aplicación que utilicemos para visualizar el documento PDF selecciona el tamaño y tipo de la fuente que se utilizará para la anotación. Los atributos de una anotación de texto son:

/Subtype Su valor siempre es **/Text**.

/Contents El texto que será visualizado; puede ocupar varios párrafos y debe ser una cadena, en el sentido anterior.

/Open Es una variable lógica que admite, por tanto, los valores **true** y **false**, que es el valor por defecto. Permite indicar si la anotación de texto aparecerá abierta o no cuando se visualice el documento.

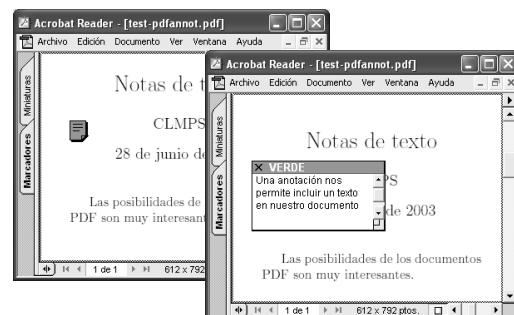
Recordemos de nuevo que el texto que incluyamos en el contenido de la nota tiene que seguir las especificaciones del lenguaje PDF, para lo cual utilizaremos los comandos **\pdfstringdef** o **\texorpdfstring**, descritos en la página 409.

Atributo	Tipo	Descripción
/Type	nombre	Tipo del objeto. Siempre es /Annot.
/Subtype	nombre	Subtipo de la anotación: /Text, /Link o /Movie.
/Rect	rectángulo	Región rectangular que especifica la localización de la anotación.
/Border	matriz	Matriz de 3 (o 4) números que especifica los radios horizontal y vertical de las esquinas y la anchura del recuadro. Si está presente el cuarto número, entonces se dibujan líneas discontinuas.
/C	matriz	El color de la anotación en el modelo RGB. Para los enlaces representa el color del recuadro. Para las anotaciones de texto indica el color de fondo del icono que representa la anotación y el color de la barra de ventana si la anotación está abierta.
/T	cadena	Etiqueta de texto que identifica la anotación y que aparece en la barra de la ventana cuando está abierta.
/M	fecha	La última fecha en que la anotación fue modificada. Preferiblemente debe estar en formato PDF, aunque los programas de visualización aceptan cualquier cadena de texto.
/H	nombre	Especifica el efecto visual al seleccionar una anotación. Los posibles valores son: /I (se muestra en colores inversos), /N (sin efecto), /O (se visualiza un cuadro), /P (se comporta como un botón).

Cuadro 6.5: Atributos comunes a todas las anotaciones de un documento PDF

EJEMPLO 6.1

```
\pdfstringdef{\TextoPDF}{Una anotación
nos permite incluir un texto en nuestro
documento}
\pdfannot width 6cm height 3cm {/Subtype /Text
/T (VERDE) /C [0 1 0] /Open false
/Contents (\TextoPDF)} Las posibilidades de los
documentos PDF son muy interesantes.
```



En la ventana de la izquierda aparece la anotación cerrada, según el valor del parámetro /Open. La ventana de la derecha muestra la anotación una vez abierta

6.2.2. Enlaces que realizan acciones

Cuando una anotación de enlace es activada se ejecuta una acción, que puede consistir en desplazarnos a otro lugar en nuestro documento PDF (es decir, un hiperenlace ordinario) o en otras

acciones más complejas, que incluso pueden realizarse consecutivamente. Los atributos de una anotación de enlace son:

/Subtype Su valor siempre es **/Link**.

/Dest Es el destino a visualizar, que se representa por un nombre o por una matriz. Debe estar presente, a menos que el atributo **/A** lo esté.

/A Es un diccionario que especifica la acción que debe emprenderse al activar el enlace y debe estar presente a menos que el atributo **/Dest** lo esté.

Las anotaciones con el atributo **/Dest** son equivalentes a los hiperenlaces obtenidos con el comando **\hyperlink** que proporciona el paquete **hyperref**, que es la opción que recomendamos para la construcción de vínculos. En este apartado vamos a ocuparnos de las anotaciones de enlace con el atributo **/A**, cuyo contenido es un diccionario.

Un diccionario es una sucesión de comandos o argumentos y sus correspondientes valores (que pueden ser a su vez comandos) válidos en lenguaje PDF y que está delimitado por los símbolos **<<** y **>>**. En general, un diccionario tiene la siguiente forma:

```
<< /Argumento_A /Valor_A  
    /Argumento_B /Valor_B  
    ...  
    /Argumento_N /Valor_N >>
```

Las posibles acciones, precedidas siempre de la clave **/S**, son:

/GoTo Tiene el mismo efecto que especificar un destino mediante la clave **/Dest** y el único atributo de esta acción es **/D** (matriz, cadena o nombre que indica el destino). Por facilidad de uso, sin embargo, recomendamos la utilización del comando **\hyperlink** cuando queramos realizar esta acción.

/GoToR Permite abrir un nuevo archivo PDF, en una página y con un factor de zoom especificados.

Los posibles atributos son: **/F** (fichero que contiene el destino), **/D** (destino) y **/NewWindow** (variable lógica que especifica si el destino debe abrirse en una nueva ventana o no).

/Launch Lanza una aplicación, usualmente abrir un fichero. Los atributos de esta acción son: **/F** (fichero), **/Win** (diccionario con los parámetros necesarios para ejecutar la aplicación) y **/NewWindow** (igual que antes, pero sólo se aplica si el destino es un archivo PDF).

/URI Enlaza con un recurso en Internet, típicamente un archivo o una dirección de una página web. Los atributos son: **/URI** (la cadena que identifica el recurso) e **/IsMap** (variable lógica). Si la variable **/IsMap** está activada, entonces hay que proporcionar en la cadena **/URI** las coordenadas del ratón.

/Named Ejecuta una acción predefinida por el visor. Acciones típicas, siempre precedidas de la clave **/N**, son las siguientes: **/FirstPage**, **/PrevPage**, **/NextPage**, **/LastPage**, **/GoForward**, **/GoBack**, **/Close** y **/Quit**. Para una lista de las principales acciones predefinidas en ACROBAT consulte el cuadro 6.4.

Algunos ejemplos de diccionarios que representan acciones son los siguientes:

```
<< /S /GoTo /D (page.9) >>
```

```
<< /S /GoToR /F (documento1.pdf) /D (page.3) >>
```

```
<< /S /Launch /F (archivo.txt) >>
<< /S /URI /URI (http://www.adobe.es/) >>
<< /S /Named /N /NextPage >>
```

En ocasiones es necesario realizar varias acciones sin que el usuario las vaya activando de una en una. Para ello disponemos del comando /Next, que requiere un diccionario como argumento, que representará la siguiente acción a realizar. Por ejemplo, supongamos que queremos incluir una acción que viaje hasta la página 5 del documento actual y la visualice a pantalla completa; el código necesario es el siguiente:

```
<< /S /GoTo /D (page.5) /Next << /S /Named /N /FullScreen >> >>
```

Lógicamente, dentro de la acción asociada al comando /Next es posible incluir otras acciones /Next, de forma que podamos realizar varias acciones de forma consecutiva.

6.2.3. Las películas y los sonidos

Las anotaciones de tipo película describen la ejecución y visualización de películas y sonidos dentro de un documento PDF. Estas anotaciones están incluidas en el documento, como los enlaces de hipertexto, pero hacen referencia a otros archivos externos que realmente contienen la película o el sonido. El área de activación puede ser invisible o rodeada como un enlace ordinario y, en el caso de películas, puede contener un póster de la misma. Los atributos de una anotación de este tipo son:

/Subtype Su valor siempre es /Movie.

/Movie Es un diccionario que contiene las características de reproducción del archivo multimedia:

/F Especifica el fichero que contiene la película o el sonido.

/Aspect Es una matriz que determina la altura y la anchura, en puntos, de la caja donde se va a reproducir el archivo. Si la caja no tiene dimensiones (la película es invisible), entonces sólo se oirá el sonido.

/Rotate Es un número entero que determina, en el caso de películas, el número de grados que se girará. Su valor por defecto es 0 y debe ser un múltiplo de 90.

/Poster Es una variable lógica que especifica, en el caso de películas, si debe visualizarse o no un póster (fotograma) de la misma.

/A Puede ser una variable lógica o un diccionario. En el primer caso determina si el archivo de la película o el sonido debe reproducirse o no cuando se hace clic en la anotación (por defecto se reproduce). En el segundo caso, es decir cuando es un diccionario, puede contener los siguientes atributos:

/ShowControls Variable lógica que determina si un panel de control debe ser visualizado.

/Mode Especifica el modo en que el archivo será reproducido. Los posibles valores son:

/Once (se reproduce un sola vez y entonces se para), /Open (se reproduce una sola vez pero el archivo sigue abierto), /Repeat (se reproduce continuamente) y /Palindrome (se reproduce continuamente, pero alternando el sentido de reproducción).

/Synchronous Es una variable lógica. Si su valor es true entonces no se devuelve el control al visor hasta que la película haya finalizado o la acción haya sido cancelada por el usuario.

/Start Es un número que especifica el tiempo, en segundos, a partir del cual debe reproducirse el archivo. Por defecto, su valor es 0.

/Duration Es un número que determina la duración, en segundos, del segmento de archivo que será reproducido.

/Rate Es un número que especifica la velocidad de reproducción. Su valor por defecto es 1.

/Volume Es un número entre 0 y 1 que determina el volumen de reproducción. Por defecto, su valor es 1.

Algunos ejemplos de anotaciones de películas y sonidos son los siguientes:

```
\pdfannot {/Rect [205 300 405 360] /Border [0 0 2] /C [1 1 0]
           /Subtype /Movie /Movie << /F (chimes.wav) >>
           /A << /ShowControls false >>}
\pdfannot {/Rect [200 150 400 350] /Subtype /Movie
           /Movie << /F (clock.avi) /Poster true >>
           /A << /ShowControls true /Rate 2 /Mode /Palindrome >>}
```

6.3. LATEX y HTML. El sistema de conversión TEX4ht

MUCHA de la información disponible en Internet está recogida en páginas HTML que son documentos «sólo texto», de reducido tamaño y fáciles de descargar desde la red. De forma parecida a lo que sucede en los documentos fuente de LATEX, en los documentos HTML, junto al texto, se incluyen las declaraciones y órdenes que el navegador necesita para componerlo y mostrarlo. Además, también son documentos estructurados con unidades como preámbulo–cabecera, cuerpo, listas, tablas, etc.

La semejanza entre los dos formatos ha propiciado la aparición de distintos sistemas de conversión entre documentos LATEX y documentos HTML que han venido a cubrir la demanda existente, sobre todo en medios científicos, para publicar en la red documentos producidos originariamente en LATEX. La tarea que realizan estos conversores es compleja, principalmente porque los recursos tipográficos en HTML son mucho más limitados que en LATEX. Esta limitación los obliga a convertir en gráficos algunos de los elementos de los documentos, como fórmulas, cajas complicadas, etc.

El sistema de conversión que proponemos es TEX4ht, creado por Eitan Gurarii [27], disponible para los sistemas operativos MS-WINDOWS y LINUX/UNIX. Con él vamos a obtener, a partir de un mismo documento TEX, el formato HTML, además de los habituales DVI, PS y PDF. En principio, no necesitaremos conocimientos del lenguaje HTML para obtener «buenos» documentos HTML directamente desde nuestros documentos fuente LATEX. Aunque, sin embargo, si pretendemos tener un control más «fino» sobre el diseño de las conversiones, deberemos conocer un poco de este lenguaje.

El sistema TEX4ht aborda el problema de la conversión de una manera original y, sobre todo, muy cercana al sistema de composición TEX: TEX4ht utiliza el compilador TEX para producir

los documentos HTML desde documentos compilados DVI. Este sistema de conversión, aunque no carece de complejidad, resulta muy eficiente porque es capaz de comprender el código de todas las estructuras que L^AT_EX entiende (incluidos los nuevos comandos y entornos definidos por los autores de manuscritos). Además, es capaz de interpretar los paquetes más habituales en la edición de documentos electrónicos con L^AT_EX (babel, graphicx, color, hyperref, etc.) de modo que no necesitaremos modificar nuestro documento fuente. Aunque, en algún caso, se pueden dar incompatibilidades entre los paquetes cargados; por ejemplo, al utilizar simultáneamente babel e hyperref, las referencias cruzadas a objetos flotantes producen errores de compilación¹. En la página 426 aportamos una solución para este caso.

6.3.1. El proceso de conversión con T_EX4ht

El sistema T_EX4ht está formado por un conjunto de ficheros para L^AT_EX, el paquete `tex4ht` acompañado de ficheros auxiliares de extensión `4ht`, dos programas ejecutables escritos en lenguaje C, `TEX4HT` y `T4HT`, y varias familias de fuentes para HTML.

El procedimiento de conversión de un documento fuente L^AT_EX a uno o varios documentos HTML, esquematizado en la figura 6.2, no está exento de complejidad, aunque en la práctica, para lanzarlo, ejecutaremos un único proceso por lotes, normalmente desde el menú de una de nuestras aplicaciones. De momento, para no perder la perspectiva de cómo se realiza la conversión, vamos describir los tres pasos necesarios para obtener el documento HTML.

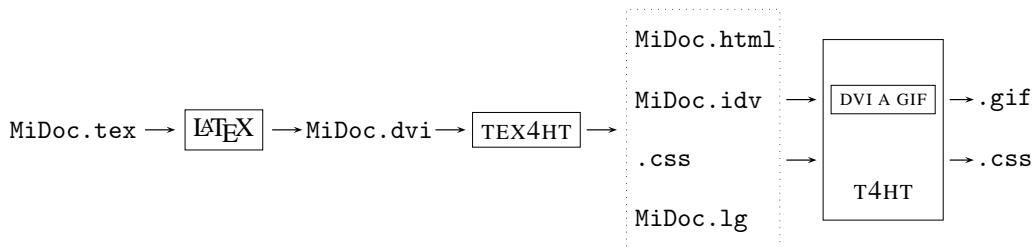


Figura 6.2: Esquema de funcionamiento de T_EX4ht

Primer paso: Se compila tres veces nuestro documento `MiDoc.tex` utilizando el paquete `tex4ht` para producir el fichero `MiDoc.dvi`, que es peculiar porque contiene información adicional para la construcción posterior de los documentos HTML y los ficheros gráficos correspondientes. La razón para hacer tres compilaciones es asegurar la correcta elaboración de referencias cruzadas, índices, tablas, etc.

Segundo paso: Se ejecuta el programa `TEX4HT` sobre `MiDoc.dvi` que produce una, o varias páginas `MiDocxx.html`, utilizando fuentes TFM de L^AT_EX y fuentes htf propias del sistema. También se crean ficheros de estilo para HTML (con extensión `css`) y un fichero `MiDoc.idv`

¹Comprobado con las versiones actuales de `hyperref` (1/6/2003 v6.74e) y las opciones de español (30/1/2001 v4.1c) alemán y francés de `babel` (1/3/2001 v3.7h).

que contiene la compilación, en formato DVI, de las partes del documento que no se han convertido en código HTML y las instrucciones para la creación de nuevas fuentes gráficas.

Tercer paso: El programa T4HT genera todos los gráficos que se usarán en el documento HTML (fuentes, gráficos, fórmulas, etc.). Para esta tarea necesita utilizar herramientas de conversión externas al sistema: el programa DVIPS para generar gráficos PostScript, el programa GHOSTSCRIPT para una primera conversión a un formato gráfico intermedio (ppm), y un programa de conversión de este formato a gif (IMAGEMAGICK en la plataforma MS-WINDOWS y NETPBM en LINUX/UNIX).

Una vez descargado e instalado el sistema **TEX4ht**, y antes de empezar a utilizarlo, es necesario que indiquemos dónde están instaladas las herramientas externas de que dispone para realizar las conversiones gráficas y cuál es la localización de las fuentes **tfm** y **htf**². Para ello, es necesario comprobar en el fichero **tex4ht.env** que las órdenes de conversión se corresponden con los programas instalados, y que las ubicaciones de programas y fuentes son las correctas.

Para facilitar la ejecución de los tres pasos del proceso de conversión, **TEX4ht** proporciona un conjunto de ficheros de procesos por lotes que realizan todos los pasos. Las dos alternativas más simples las proporcionan los siguientes, que representamos junto con la sintaxis para su ejecución.

```
htlatex "MiDoc.tex" "OpcionesPaquete" "OpcionesTeX4ht" "OpcionesT4ht"  
ht latex "MiDoc.tex" "OpcionesTeX4ht" "OpcionesT4ht"
```

El proceso HTLATEX se usa directamente sobre nuestro documento **MiDoc.tex** sin ninguna modificación, ya que el programa se encarga de incluir en las tres compilaciones con **LATEX**, la línea **\usepackage**[*OpcionesPaquete*]{tex4ht}

Una vez que ha concluido la primera etapa de compilación, HTLATEX realiza los otros dos pasos incluyendo en cada ejecución las opciones declaradas, *OpcionesTeX4ht* y *OpcionesT4ht*.

Para el proceso HT, necesitamos incluir la línea

```
\usepackage[OpcionesPaquete]{tex4ht}
```

en nuestro documento **MiDoc.tex**, ya que HT va a lanzar tres compilaciones de **LATEX** (que es su primer argumento) directamente sobre el fichero **MiDoc.tex** sin incluir ninguna orden adicional. Los otros dos pasos se ejecutan con las opciones declaradas.

En el ejemplo 6.2 podemos observar el resultado de convertir un documento simple de **LATEX**.

Los usuarios de MS-WINDOWS tienen a su disposición otras dos herramientas que facilitan el uso del sistema **TEX4ht**. El programa HTRUN, creado por Philip A. Viton (véase el apartado PARA SABER MÁS de este capítulo), que facilita la selección de opciones y permite la conversión de documentos a formatos extendidos de HTML (XHTML, MATHML, MOZILLA, etc.). Y la interfaz gráfica TEXCONVERTER [43], creada por Steve Mayer, que, entre otras cosas, permite ejecutar los ficheros de proceso por lotes de **TEX4ht**, con la posibilidad de seleccionar cómodamente las distintas opciones. También puede trabajar con HTRUN. En la figura 6.3 se muestran las pantallas de conversión y opciones de este programa.

²Si realizó la instalación desde el CD-ROM, sin modificar las direcciones asignadas por defecto a los programas, el sistema debe estar operativo y no será necesario efectuar esta operación.

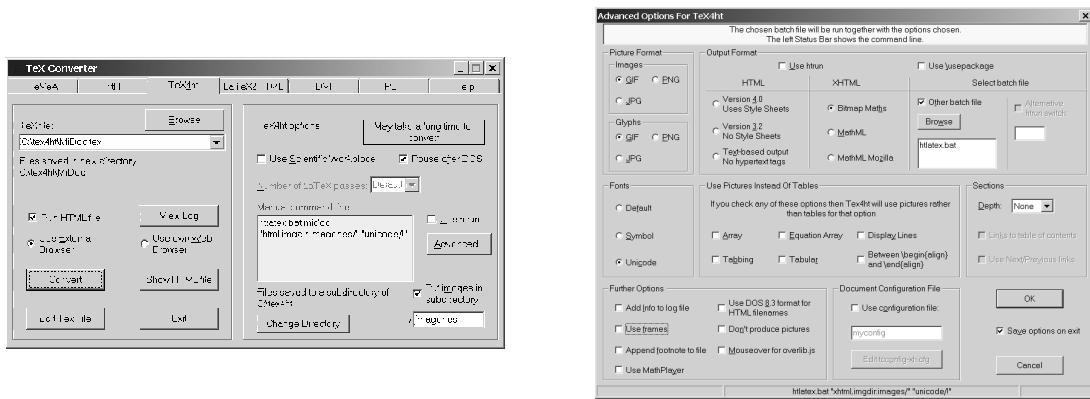


Figura 6.3: Ventanas de ejecución y selección de opciones de $\text{\TeX}4ht$ para MS-WINDOWS

6.3.2. Opciones del sistema $\text{\TeX}4ht$

En este apartado vamos a describir las opciones de cada una de las tres etapas del proceso de conversión utilizando $\text{\TeX}4ht$.

Opciones del paquete

El paquete tex4ht , utilizado en la primera etapa de la conversión para compilar el documento fuente con \LaTeX , admite las siguientes opciones, que permiten seleccionar algunas de las propiedades del documento HTML que se desea construir.

Opciones de formato

html Indica que la salida debe estar en lenguaje HTML, en la versión 4.0 (por defecto).

3.2 Se usará la versión 3.2 del lenguaje HTML. Debe emplearse esta opción si se quiere que los ficheros se visualicen correctamente al utilizar versiones antiguas de los navegadores.

0.0 Convierte el documento en una página de texto sin enlaces.

htm Produce los ficheros con nombres cortos, en formato DOS8.3, y extensión HTM.

no_|no[^] Indica que no se van a convertir los subíndices (resp. superíndices) de \LaTeX en subíndices (resp. superíndices) de HTML.

uni-html Utiliza fuentes unicode y convierte todas las fórmulas en gráficos.

xhtml La salida se construirá utilizando alguna de las extensiones de HTML (MATHML, MOZILLA, etc.). Por defecto, si no se añade la correspondiente opción, las fórmulas matemáticas que no sabe convertir las incluye como gráficos.

mathml|mozilla Se utilizan conjuntamente con la opción **xhtml** para representar las fórmulas empleando, respectivamente, las extensiones MATHML o MOZILLA.

docbook|docbook-mml También se utilizan conjuntamente con la opción **xhtml** para crear documentos de la extensión «docbook» de HTML con las fórmulas representadas en formato gráfico, o en formato MATHML si se utiliza **docbook-mml**.

De L^AT_EX a HTML con T_EX4ht

Indice

- 1 Fórmulas
- 2 Listas
- 3 Cuadros

1. Fórmulas

T_EX4ht intenta convertir las fórmulas incluidas en un párrafo utilizando fuentes HTML, $x^2+y^2=1$, mientras que las resaltadas se convierten en gráficos

$$\int_0^1 \frac{\sin x}{x} dx$$

2. Listas

- Un primer ítem
- El segundo, con una enumeración
 - 1. Primer subítem
 - 2. segundo subítem

3. Cuadros

a	b	c
A	B	C

EJEMPLO 6.2

```
%% Documento MiDoc.tex
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[spanish]{babel}
\usepackage[graphicx]
\title{De LATEX a HTML con TEX4ht}
\author{} \date{}
\begin{document}
\maketitle
\tableofcontents
\section{Fórmulas}
\TeX4ht intenta convertir las fórmulas incluidas en un párrafo utilizando fuentes HTML,  $x^2+y^2=1$ , mientras que las resaltadas se convierten en gráficos
$$\int_0^1 \frac{\sin x}{x} dx
\section{Listas}
\begin{itemize}
\item Un primer ítem
\item El segundo, con una enumeración
\begin{list-item}
\begin{list-item}
1. Primer subítem
\end{list-item}
2. segundo subítem
\end{list-item}
\end{itemize}
\begin{itemize}
\item Un primer subítem
\item Un segundo subítem
\end{itemize}
\end{itemize}
\end{itemize}
\begin{itemize}
\item Un primer ítem
\item Un segundo, con una enumeración
\begin{list-item}
\begin{list-item}
1. Primer subítem
\end{list-item}
2. segundo subítem
\end{list-item}
\end{list-item}
\end{itemize}
\begin{table border="1">
| a | b | c |
| A | B | C |


\begin{center}
\begin{tabular}{|c|c|c|} \hline
a & b & c \\ \hline
A & B & C \\ \hline
\end{tabular}
\end{center}
\end{document}
```

Conversión simple de L^AT_EX a HTML obtenida con: htlatex MiDoc.tex "html" "iso8859/1"

tei|tei-mml Crea documentos de la extensión «tei» de HTML si se usa con la opción `xhtml`, con las fórmulas representadas en formato gráfico (tei) o en MATHML (tei-mml).

word Crea documentos HTML para ser manejados por Microsoft-Word.

Opciones de división y enlace

1, 2, 3, o 4 Representa el nivel de división del documento en varias páginas HTML convenientemente enlazadas para poder navegar por ellas siguiendo la estructura del documento: 1 para una página por cada capítulo, 2 para una página por sección, 3 para subsecciones y 4 para subsubsecciones. En el ejemplo 6.3 se pueden ver dos de las páginas que resultan al convertir un documento con la opción 2. Obsérvese en este ejemplo cómo han aparecido dos barras de navegación en la página de la sección, aunque dichas barras están construidas en inglés. Las barras de navegación se configuran fácilmente tal y como veremos en el siguiente apartado (véanse los ejemplos 6.10 y 6.11).

section+ Crea enlaces en los títulos de las unidades de estructura (secciones, subsecciones, etc.) para regresar al índice general.

next En documentos divididos en varias páginas, define los enlaces «next» (página siguiente) y «previous» (página anterior) en todas las páginas para poder navegar por el documento.

frames Con esta opción el documento se divide en distintas páginas HTML a las que se accede utilizando una página descompuesta en «frames», marcos o ventanas, donde se representan simultáneamente varias páginas HTML. Por defecto, con esta opción el índice de contenidos aparece en el marco lateral izquierdo, y a la derecha aparece la ventana principal en la que se van mostrando las distintas páginas del documento. Esta situación se puede modificar utilizando los comandos de configuración de `TeX4ht`.

fn-in Se usa para que las notas a pie de página aparezcan al final de cada página convenientemente enlazadas. Por defecto, el sistema crea una página independiente para cada nota con los correspondientes enlaces. En la versión analizada de `TeX4ht`, su uso inhabilita la construcción de «frames» sin ofrecer mensajes de error.

Opciones para manipular gráficos

jpg, png Por defecto `TeX4ht` produce los gráficos a incluir en los ficheros HTML en formato GIF; con estas opciones podemos indicarle que produzca los gráficos en formato JPG o PNG.

img:*NombreSubdir* Si está presente esta opción, el conversor genera todos los gráficos en el subdirectorio *NombreSubdir*, que será creado en caso de no existir.

pic-array, pic-eqnarray, pic-tabling, pic-tabular, pic-align Cada una de estas opciones indica al conversor que el contenido de los entornos del mismo nombre debe convertirse en gráficos que son incluidos en el documento HTML.

Fichero de configuración

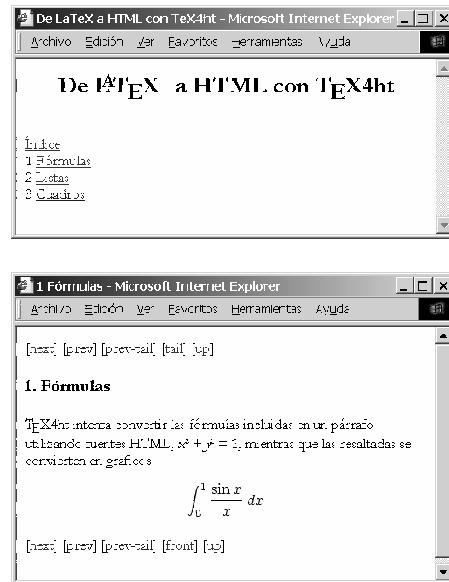
FicheroConfig El paquete `tex4ht` también admite como opción el nombre de un fichero de configuración (*FicheroConfig*). En el siguiente apartado describimos su contenido y su estructura.

EJEMPLO 6.3

```
% Documento MiDoc.tex
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[spanish]{babel}
\usepackage{graphicx}
\title{De \LaTeX{} a HTML con \TeX4ht}
\author{}
\date{}
\begin{document}
\maketitle
\tableofcontents
\section{Fórmulas}
\TeX4ht intenta
...

$$\int_0^1 \frac{\sin x}{x} dx$$

\section{Listas}
...
\section{Cuadros}
...
\end{document}
```



Conversión de LATEX a HTML usando la opción 2 de tex4ht con: `htlatex MiDoc "html,2" "iso8859/!"`

Opciones de los programas TEX4HT y T4HT

Los programas TEX4HT y T4HT, utilizados en la segunda y tercera etapas de la conversión para construir las páginas HTML, también admiten diferentes opciones como parámetros de ejecución.

Opciones de TEX4HT

-g.jpg, -g.png Indica que las fuentes gráficas generadas en la conversión deben guardarse en el formato JPG o PNG. Por defecto, si no se usa esta opción, se guardan en formato GIF.

SubDirFuentes! Indica el subdirectorio del directorio de fuentes del conversor, *SubDirFuentes*, que contiene las definiciones de las fuentes de hipertexto que usaremos en el documento HTML. El sistema proporciona estos ficheros para los siguientes conjuntos de fuentes, distribuidos en subdirectorios del mismo nombre: *dbscs*, *iso8859*, *symbol* y *unicode*. Por ejemplo, si queremos convertir documentos escritos en *latin1* podemos utilizar las fuentes *iso8859* o *unicode* para evitar problemas con las letras acentuadas y sobre todo con las ligaduras de tipos que produce LATEX.

Opción para T4HT

-p Con esta opción el programa T4HT omitirá la generación de gráficos. Esta opción es muy útil porque la generación de gráficos es el proceso más lento de la conversión de documentos. Así, mientras realizamos modificaciones en el documento que no afectan a las fórmulas y gráficos podemos hacer conversiones sin gráficos, dejando la reconstrucción de los mismos para la última conversión.

6.3.3. Ficheros de configuración

Un fichero de configuración para el estilo `tex4ht` es un fichero «sólo texto» con extensión `cfg` (por ejemplo, `MiDocHTML.cfg`), que contiene una lista adicional de opciones del paquete, los comandos propios de `TEX4ht` con los que se controlan las modificaciones del estilo de las páginas HTML a construir, y texto para añadir al inicio del documento. Para utilizarlo durante la conversión sólo tenemos que pasar su nombre (`MiDocHTML`) como opción del paquete.

Los ficheros de configuración tienen la estructura de un entorno, comienzan obligatoriamente con el comando `\Preamble` y terminan con `\EndPreamble`. Están divididos en dos partes separadas por la declaración `\begin{document}`, también obligatoria. Esta declaración es independiente de la que debe aparecer en el documento `LATEX` que se va a convertir. La primera parte contiene las modificaciones del estilo de conversión. La segunda contiene texto y comandos para añadir al inicio del documento fuente.

```
%%Fichero de configuración MiDocHTML.cfg
\Preamble{...}
% Nuestros comandos para modificar el estilo de conversión del documento
\begin{document}
% Comandos y texto para añadir al documento fuente
\EndPreamble
```

El argumento de `\Preamble` puede contener una lista de opciones del paquete `tex4ht` que deseemos usar con este fichero de configuración. Esta lista es suplementaria a la que acompaña al paquete.

El programa `TEXCONVERTER` nos proporciona algunos ficheros de configuración predeterminados que se pueden complementar con ficheros que utilizan el mismo nombre que el documento fuente con extensiones `cfg` y `cfh`. Éstos sólo contendrían órdenes adicionales que se incluirían, respectivamente, antes y después de la línea `\begin{document}` durante la compilación.

6.3.4. Comandos y entornos básicos de `TEX4ht`

El paquete `tex4ht` proporciona un conjunto de comandos que permiten modificar el estilo de la conversión e insertar código HTML, enlaces, gráficos, etc. En este apartado describiremos algunos de ellos con la intención de mostrar cómo se modifican los parámetros que se usan por defecto. En particular, daremos soluciones a los problemas de incompatibilidad entre `babel` e `hyperref` antes señalados, y mostraremos como cómo construir la barra de navegación que aparece en el ejemplo 6.3.

Para trabajar manteniendo la filosofía, que también comparten `LATEX` y `HTML`, de escribir en ficheros de estilo las modificaciones que afectan a todo el documento, es muy recomendable trabajar con ficheros de configuración en los que situar los comandos de `tex4ht`. Una razón importante para intentar no introducir este código en el documento `LATEX` es la de no tener que manejar diferentes versiones de un mismo documento en función del tipo de salida que deseemos obtener, facilitando, de ese modo, las tareas de edición y mantenimiento del mismo.

En el manual del programa [27] se puede encontrar una descripción completa de todos los comandos y entornos de `tex4ht`. En este apartado sólo analizaremos algunos de ellos.

Tenemos que advertir que la sintaxis de los entornos de `tex4ht`, que ha sido diseñado para trabajar también con ficheros escritos en otros formatos, respeta la sintaxis original de `TEX`, al estar limitados por dos comandos de la forma `\Entorno` y `\EndEntorno` en lugar de los habituales `\begin{Entorno}` y `\end{Entorno}` de `LATEX`.

Código HTML

Los dos comandos siguientes permiten incluir código HTML

<code>\HCode{CódigoHTML}</code>	<code>\HChar{Número}</code>
---------------------------------	-----------------------------

El argumento *CódigoHTML* es cualquier texto en código del lenguaje HTML que no incluya caracteres reservados de `LATEX` (véase el ejemplo 6.4). Para usar estos caracteres podemos utilizar su representación en `LATEX` o el comando `\HChar`, que inserta literalmente el carácter que, en la página de códigos utilizada, tiene por código el *Número* puesto como argumento.

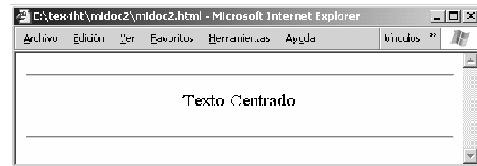
Por ejemplo, uno de los caracteres más incómodos para incluir es, sin duda, el %. Para usarlo en una línea como `<table width="80%">`, podemos escribirlo con cualquiera de las dos expresiones siguientes:

```
\HCode{<table width="80\%">}
\HCode{<table width="80\%>}\HChar{37}\HCode{">}
```

Importante: Los usuarios del paquete `babel` con la opción `spanish` no deben olvidar desactivar algunos métodos taquigráficos para poder utilizar sin problemas de compilación símbolos como <, >, ", '..., tan habituales en HTML (véase la sección 1.2.3).

EJEMPLO 6.4

```
\HCode{<HR>}
\begin{center}
    Texto Centrado
\end{center}
\HCode{<hr>}
```



Insertando código HTML

La presentación del contenido de las páginas HTML se puede asociar a páginas de estilo en cada (`css`) de forma parecida a como los ficheros `LATEX` se compilan utilizando paquetes. `TEX4ht` produce un fichero de estilo `css` asociado a cada documento convertido en HTML utilizando la versión 4.0 (opción por defecto). El comando

<code>\Css{CódigoEstilo}</code>

sirve para añadir el argumento *CódigoEstilo* al fichero de estilo. Por ejemplo, situando en el preámbulo del fichero de configuración la línea

```
\Css{.cuadroazul{ color:white; background-color:blue;}}
```

añadiremos el método *cuadroazul* a nuestro fichero de estilo y podremos referirnos a él cada vez que queramos obtener elementos sobre fondo azul con letras blancas (véase el ejemplo 6.9).

```
\Preamble{html}
  \newcommand{\href}[2]{\Link[#1]{}{}#2\EndLink}
  \newcommand{\hyperlink}[2]{\Link[]{}{}#2\EndLink}
  \newcommand{\hypertarget}[2]{\Link[]{}{}#1#2\EndLink}
  \begin{document}
\EndPreamble
```

Figura 6.4: Ejemplo de fichero de configuración para evitar incompatibilidades

Enlaces

Al utilizar el paquete `tex4ht` se crean, de forma automática, hiperenlaces entre las unidades de estructura y el índice general, entre las referencias cruzadas, entre la bibliografía y sus citas, entre las marcas de las notas a pie de página y estas notas, etc.

Para construir hiperenlaces adicionales, `tex4ht` dispone de un solo entorno con el que se pueden crear tanto los enlaces como los puntos de destino.

```
\Link[URL]{DestinoEnlace}{EtiquetaEnlace}
TextoEnlace
\EndLink{}
```

Para definir el punto de destino *EtiquetaEnlace* en *TextoEnlace*, hacia donde pueden apuntar futuros hiperenlaces, se escribe

```
\Link[]{}{EtiquetaEnlace}TextoEnlace\EndLink{}
```

Para crear un enlace desde *TextoEnlace* hacia el punto de destino *DestinoEnlace*, cuando el enlace y el destino están en la misma página, se usa

```
\Link[]{}{DestinoEnlace}{}TextoEnlace\EndLink{}
```

Por último para crear un hiperenlace desde *TextoEnlace* hacia otras páginas de Internet o hacia otra aplicación, se utilizan

```
\Link[URL]{}{}TextoEnlace\EndLink{}, o
```

```
\Link[URL]{DestinoEnlace}{}TextoEnlace\EndLink{}
```

para apuntar hacia URL o URL#DestinoEnlace, respectivamente.

Para escribir los caracteres reservados de L^AT_EX ~, _, # y % en las URL, se pueden utilizar los comandos `\string ~`, `\string _`, `\#` y `\%`, respectivamente.

La estrategia más habitual para incluir hiperenlaces en nuestros documentos es utilizar el paquete `hyperref` (véase la sección 8.3). Este paquete es compatible con `tex4ht`, y su empleo es recomendable si se quiere disponer de distintos formatos de salida a partir de un mismo documento fuente. Sin embargo, en las versiones analizadas de los paquetes, se presenta un problema de compatibilidad al usar conjuntamente `hyperref`, `babel` y `tex4ht`. Una solución a este problema, sin necesidad de incluir código adicional a nuestro documento fuente, consiste en no cargar el paquete `hyperref`, y usar para la conversión un fichero de configuración donde se incluyen las definiciones de los comandos utilizados del paquete `hyperref`. La figura 6.4 muestra un ejemplo de un fichero de configuración.

Gráficos

El sistema `TEX4ht` entiende los comandos de inclusión o generación de gráficos de los paquetes más usuales de `LATEX`, por ejemplo `graphicx` o `pstricks`. Es capaz de convertir los gráficos utilizados en nuestro documento a los formatos habituales en Internet(GIF, JPG o PNG), insertándolos posteriormente en el documento HTML.

Utilizando comandos específicos del paquete `tex4ht`, se pueden incorporar gráficos existentes en la red, o convertir en gráfico cualquier parte de nuestro documento. Para incorporar gráficos desde Internet, disponemos del comando

```
\Picture[TextoAlternativo]{FicheroGrafico-Atributos}
```

donde *FicheroGrafico* es la dirección de la red del fichero gráfico a incorporar y *Atributos* son las propiedades HTML del gráfico. El argumento optativo *TextoAlternativo* es el texto que aparecerá al ver el documento con un navegador de texto.

EJEMPLO 6.5

Escudo de la Universidad de Murcia

```
\begin{center}
\Picture[umu]{
  http://www.um.es/graficos/escudo.gif %
  width="100pt"
} \end{center}
```



Incorporando gráficos desde Internet

El paquete también permite convertir en gráfico cualquier parte del documento; para realizar esta tarea se dispone del siguiente entorno:

```
\Picture+ [TextoAlternativo] {NombreGráfico}
Texto a convertir
\EndPicture
```

donde *NombreGráfico* es el nombre con el que se guardará el gráfico creado.

EJEMPLO 6.6

```
\Picture+[saludo]{logoSaludo}
\fboxrule=4pt \fboxsep=4pt
\fcolorbox{blue}{yellow}{\color{blue}Hola}
\EndPicture
```

Hola

Gráfico logoSaludo.gif generado por el entorno Picture

Nuevas páginas HTML

El siguiente entorno, útil para hacer anotaciones extensas, crea una nueva página HTML con sus correspondientes enlaces, desde y hacia nuestro documento.

```
\HPage{TextoEnlace}
Texto
\ExitHPage{TextoEnlaceRetorno}
Más Texto
\EndHPage{}
```

crea una página HTML a la que se accede desde nuestro documento en *TextoEnlace*, y cuyo contenido es el que corresponde al *Texto* comprendido entre \HPage y \EndHPage. En cualquier parte de la nueva página podemos incluir el comando \ExitHPage que crea enlaces de retorno en *TextoEnlaceRetorno*, con destino hacia el documento principal en *TextoEnlace*.

Índice general

T_EX4ht proporciona una extensión del comando \tableofcontents que permite indicar las unidades de estructura que van a aparecer en el índice general

```
\tableofcontents [unidad1,unidad2...]
```

Por ejemplo, al incluir \tableofcontents [chapter,likechapter,section,likesection], se crea el índice general con las unidades \chapter, \chapter*, \section y \section*, así como todas las unidades creadas a partir de éstas.

El comando

```
\TocAt{EnUnidad,unidad1,unidad2...}
```

colocado en el preámbulo de un fichero de estilo, produce un índice general de cada unidad de estructura declarada por *EnUnidad*, que contiene los títulos de las unidades que aparecen a continuación. Por ejemplo,

```
\TocAt{section,subsection,likesubsection,
      subsubsection,likesubsubsection}
```

produce un índice en cada sección que incluye las subsecciones y subsubsecciones (con o sin *).

Modificando la conversión: «hooks»

El paquete tex4ht encapsula los comandos originales de L_AT_EX, enganchándoles «hooks» que son listas de órdenes para ejecutar durante la conversión, antes y después del comando. Así, modificando la configuración de estos «hooks» se puede modificar el estilo de la conversión a HTML.

Los «hooks» se modifican con el comando

```
\Configure{NombreHook}{Parámetros1}{Parámetros2}...{ParámetrosN}
```

donde *NombreHook* es el nombre del «hook» y *Parámetros1...* son las listas de órdenes que tiene asociado. El número de parámetros de cada «hook» está predefinido y depende de la tarea que tiene encomendada.

A continuación vamos a relatar alguno de los «hook» correspondientes a estructuras básicas de L_AT_EX incluyendo algunos ejemplos de modificaciones.

Fórmulas Para reproducir fórmulas matemáticas definidas en entornos `\(...\)` y `\[...\]` TEX4ht las convierte en gráficos y las incluye en el documento, mientras que las definidas entre `$...$` se construyen con símbolos de las fuentes HTML.

Para modificar este comportamiento, por ejemplo, para que estas últimas fórmulas también se conviertan en gráficos, incluiremos en el preámbulo del fichero de configuración, la línea:

```
\Configure{$}{\Picture+{} }{\EndPicture}{}{}
```

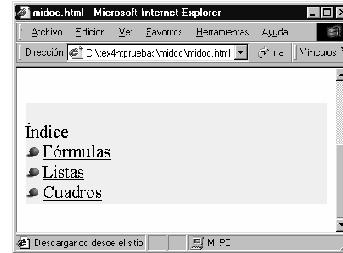
Índice general Para cambiar el aspecto del índice general y de otros índices disponemos, entre otros, de los siguientes comandos de configuración:

```
\Configure{tableofcontents}{AntesIndice}{AlFinalIndice}{DespuesIndice}%
{AntesParNoIndent}{AntesParIndent}
\ConfigureToc{Unidad}{AntesMarca}{AntesTitulo}{AntesNumPag}{AlFinal}
```

Al utilizarlos, es necesario reconstruir con código HTML toda la estructura que queramos que tengan los índices. En el ejemplo 6.7 utilizamos estos dos comandos y una redefinición del comando `\thesection` para modificar la representación del índice general.

EJEMPLO 6.7

```
%% Fichero de configuración MiDocHTML.cfg
\renewcommand{\thesection}{}
\Configure{tableofcontents}%
{\HCode{<div style="background-color:#CCFFFF">}}%
{\maketitle}{\HCode{<br>}}%
{\HCode{</div>}}{\HCode{<br>}}{}%
\ConfigureToc{section}%
{\par\Picture{bolaVerde.gif width="20pt"}%
{\HCode{<span style="text-decoration:none" >}}%
{}{\HCode{</span>}}}
```



Modificación del índice general

Listas y entornos Las listas son objetos habituales en HTML y, en consecuencia, la conversión de las listas usuales de LATEX no ofrece ningún problema. El comando

```
\ConfigureList{NombreLista}{AntesLista}{DespuesLista}%
{AntesEtiqueta}{DespuesEtiqueta}
```

puede utilizarse para redefinir los «hooks» correspondientes a cada uno de los entornos que han sido construidos a partir de los entornos `list` y `trivlist`. Por ejemplo, `description`, `itemize`, `enumerate`, `verse`, `quotation` y `quote`, así como también `center`, `flushleft` y `flushright`.

Sobre la base de este comando, `tex4ht` también ofrece el siguiente, que permite configurar las acciones a realizar antes y después de cualquier entorno:

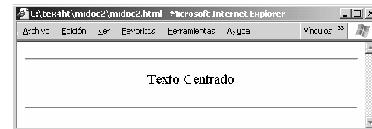
```
\ConfigureEnv{NombreEntorno}{AntesEntorno}{DespuesEntorno}{}{}
```

En el ejemplo 6.8 se modifica la configuración del «hook» `center` para incluir líneas horizontales antes y después de cada uno de los entornos `center` que aparezcan en el documento fuente (véase también el ejemplo 6.4).

Tablas Las tablas son otros de los objetos habituales en HTML, y **TeX4ht** realiza las conversiones de los entornos tabulados de **LATEX** (`tabular`, `array`, etc.) sin ninguna dificultad. Cada uno de esos entornos tiene asociado un «hook» con el mismo nombre. La configuración de estos «hook» se realiza con el siguiente comando:

EJEMPLO 6.8

```
%% MiDocHTML.cfg
\ConfigureEnv{center}{\HCode{<HR>}}{\HCode{<HR>}}{}{%
%MiDoc.tex
\begin{center}    Texto Centrado    \end{center}}
```



Configuración de `center`

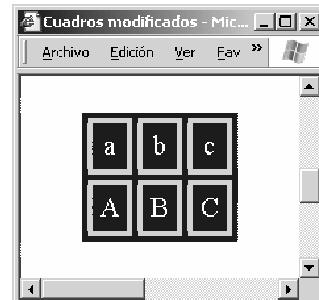
```
\Configure{EntTabla}{AntesTabla}{DespuésTabla}{AntesFila} %
{DespuésFila}{AntesCelda}{DespuésCelda}
```

donde *EntTabla* es el nombre del entorno (`tabular`, `array`, etc.) y los demás argumentos declaran las acciones y el código a incluir antes y después de cada elemento de la tabla.

En el ejemplo 6.9 se observa cómo hemos tenido que reconstruir la estructura de tabla con código HTML para reconfigurar el entorno `tabular` de manera que produzca tablas con fondo azul, letras blancas y celdas enmarcadas.

EJEMPLO 6.9

```
%%Fichero Configuración MiDocHTML.cfg
\Configure{tabular}
{\HCode{<table class="cuadroazul">}}
{\HCode{</table>}}{\HCode{<tr>}}{\HCode{</tr>}}
{\HCode{<td class="bordeCelda">}}
{\HCode{</td>}}
\ Css{.cuadroazul{color:white;background-color:blue;}}
\ Css{.bordeCelda{ border:solid 4px; }}
%Fichero MiDoc.tex
\begin{tabular}{lll}
a&b&c \\
A & B & C
\end{tabular}
```



Configuración de las tablas

Barras de navegación Como ya hemos comentado en la página 422, **tex4ht** genera barras de navegación en cada una de las páginas en las que se descompone el documento HTML cuando se selecciona una de las opciones 1, 2... del paquete. El comando interno del paquete que las genera tiene asociado el «hook» `crosslinks`, que se configura con:

```
\Configure{crosslinks}{AntesEnlace}{DespuésEnlace}{EnlaceSiguiente} %
{EnlaceAnterior}{EnlaceFinalAnterior}{EnlaceArriba}{EnlaceAbajo}{EnlaceInicio}
\Configure{crosslinks+}{AntesBarraNavSup}{DespuésBarraNavSup} %
{AntesBarraNavInf}{DespuésBarraNavInf}
```

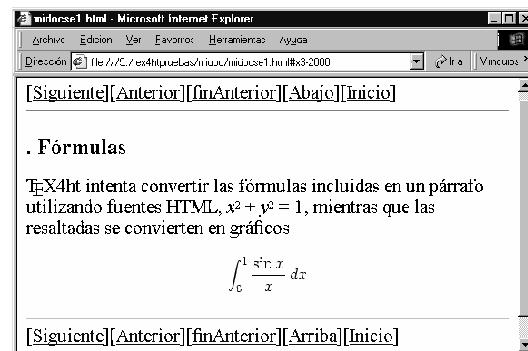
Por defecto, los valores de su configuración son

```
\Configure{crosslinks}{[]}{[]}{next}{prev}{prev-tail}{tail}{up}
\Configure{crosslinks+}{[]}{[]}{[]}{}
```

EJEMPLO 6.10

```
%% Fichero Configuración MiDocHTML.cfg

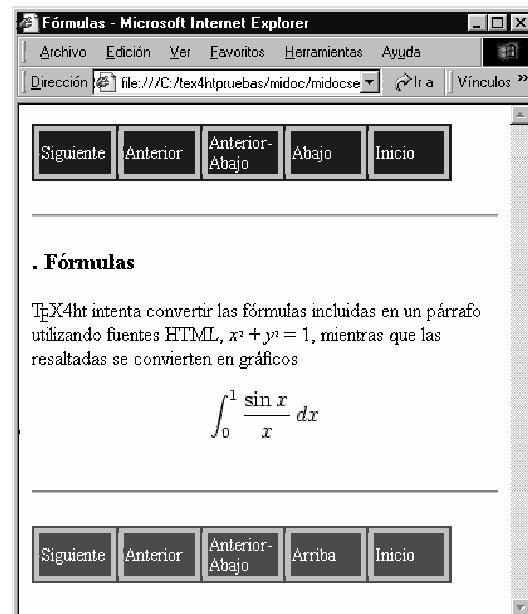
\Configure{crosslinks}{[]}{[]}{Siguiente}{Anterior}%
    {finAnterior}{Arriba}{Abajo}{Inicio}
\Configure{crosslinks+}{[]}{\HCode{<hr>}}%
    {\HCode{<hr>}}{}
```



Configuración de la barra de navegación

EJEMPLO 6.11

```
%% Fichero Configuración MiDocHTML.cfg
\newcommand{\boxmenu}[1]%
{\HCode{<span style="text-decoration:none; color:white;font-size:14" >}#1\HCode{</span>}}
\Configure{crosslinks}{\HCode{<td style="border:solid 4px;" width="20"}\HChar{37}\HCode{>}}{\HCode{</td>}}%
{\boxmenu{Siguiente}}{\boxmenu{Anterior}}%
{\boxmenu{Anterior-Abajo}}{\boxmenu{Arriba}}%
{\boxmenu{Abajo}}{\boxmenu{Inicio}}
\Configure{crosslinks+}{\HCode{<table % style="background-color:blue" width="90%" \HChar{37}\HCode{>}<tr>}}{\HCode{</tr></table>% <BR><HR>}}{\HCode{<br><HR><br>}}{\HCode{<table % style="background-color:green" width="90%" \HChar{37}\HCode{>}<tr>}}{\HCode{</tr></table>}}
```



En el ejemplo 6.10 se realiza una primera reconfiguración para crear las barras de navegación en español separándolas del contenido de las páginas con líneas horizontales. Ésta es una primera solución al problema observado en el ejemplo 6.3. Con un poco más de código HTML podemos obtener resultados más vistosos, como los presentados en el ejemplo 6.11.

PARA SABER MÁS

- Creemos que con la descripción que hemos realizado del sistema de conversión $\text{\TeX}4\text{ht}$ se pueden conseguir buenos documentos HTML a partir de documentos fuente en \LaTeX , pero somos conscientes de haber dejado sin explorar otras interesantes posibilidades del sistema como las que relacionamos a continuación:
- creación de nuevas unidades de estructura;
 - control de la división de los documentos;
 - definición y configuración de nuevos «hooks»;
 - selección de las fuentes;
 - programación de la ejecución de los comandos, en función de las opciones elegidas, utilizando condicionales específicos del paquete.

Los lectores interesados en utilizarlas encontrarán la descripción de los comandos y entornos necesarios, y muchos ejemplos que muestran su comportamiento, en el propio manual de $\text{\TeX}4\text{ht}$ [27], en el estupendo libro de M. Goosen y S. Rahtz (con E. Gurari, R. Moore y R. Sutor) [22] y en el artículo de F. Popineau [55].

- El manual de $\text{\TeX}4\text{ht}$, *TeX4ht: LaTeX and TeX for Hypertext*, está accesible en el sitio de internet de su autor, Eitan M. Gurari. La dirección exacta del mismo es
- <http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn.html>
- En la página de P.A. Viton se encuentra el documento *TeX4HT under MiKTeX*, con información adicional para el uso de $\text{\TeX}4\text{ht}$ con MiK \TeX :
- <http://facweb.arch.ohio-state.edu/pviton/support/tex4ht.html>
- El interface de conversión TEXCONVERTER creado por S.Mayer, puede descargarse de su página personal; la dirección exacta es:
- <http://www.mayer.dial.pipex.com/tex.htm>
- Aquí también encontrará más documentación del programa, otros ejemplos y numerosos enlaces de interés.
- Existen otros conversores de \LaTeX a HTML que trabajan directamente sobre el documento fuente \LaTeX , y consisten en un conjunto de reglas que además de trasladar la estructura del documento intentan simular el resultado de la composición con \LaTeX . De entre estos tipos de conversores destacamos $\text{\LaTeX}2\text{HTML}$, creado por N. Drakos (1993) y revisado por R. Moore (desde 1997) [18], y HeVeA, creado por L. Maraouget [41].

Presentaciones

Hoy en día es habitual que las salas de conferencias dispongan de medios técnicos suficientes para poder realizar charlas y presentaciones atractivas: pantallas, proyectores de diapositivas, retroproyectores de transparencias, proyectores digitales, etc. Existen diversas herramientas que pueden ayudarnos en la elaboración de transparencias y presentaciones, como MAGICPOINT (software gratuito para las plataformas LINUX/UNIX) o POWERPOINT (software comercial para los sistemas MS-WINDOWS). Pero los resultados obtenidos con cualquiera de ellos no son muy satisfactorios si las presentaciones deben contener gran cantidad de fórmulas, algo habitual en las conferencias de carácter científico-técnico.

La elaboración y publicación de documentos electrónicos están actualmente ligadas al formato PDF, por las razones ya expuestas en el apartado I.1.3. Consecuentemente, sería deseable disponer de herramientas que nos permitieran realizar presentaciones utilizando este tipo de archivos y que tuviesen al menos las siguientes cualidades:

- Que proporcionen métodos sencillos para la elaboración de presentaciones.
- Que ayuden en la mezcla de fórmulas con texto e imágenes; tarea que no realizan adecuadamente las herramientas WYSIWYG actuales.
- Que exploten la independencia de la plataforma que posee el sistema L^AT_EX, de forma que la presentación sea también independiente de la plataforma.
- Que permitan reutilizar el código L^AT_EX de los documentos disponibles, relativos al trabajo que se va a exponer.

En este capítulo vamos a presentar dos paquetes para la realización de presentaciones con L^AT_EX. Con estos dos paquetes pretendemos responder a dos situaciones de partida diferentes para las que recomendamos su uso. En último extremo, dejamos al lector la elección del sistema a utilizar en función de sus gustos y sus necesidades.

En la primera sección presentamos el paquete web, con el cual es muy fácil preparar una presentación a partir de un documento L^AT_EX que contiene (salvo pequeñas modificaciones) la información que queremos exponer. Este paquete se complementa con la utilidad externa PPOWER4 que permite animar la presentación construida. En la segunda sección analizamos esta utilidad y los paquetes pause y background que permiten controlar las animaciones.

El otro procedimiento para crear presentaciones, analizado en la tercera sección, lo proporciona la clase prosper. Es muy útil cuando queremos construir una presentación fijando el contenido

pdftex	dvips	dvipsone
dviwindo	dvipdfm	textures

Cuadro 7.1: Controladores admitidos por el paquete web

y la animación de cada pantalla. A diferencia del paquete anterior, con *prosper* las pantallas se construyen sobre la base de los recursos gráficos del potente lenguaje PostScript; por esta razón, L^AT_EX construye primero la presentación (incluyendo las posibles animaciones) en formato PS y después ésta se convierte al formato PDF.

7.1. Presentaciones con el paquete web

ESTA sección describe el uso del paquete *web* [63], desarrollado por D. P. Story, que está concebido para ser utilizado cuando la salida es un documento PDF. No obstante, las capacidades completas de este paquete sólo se pueden aprovechar si el documento final PDF es visualizado con el programa ACROBAT READER.

El propósito del paquete *web* es crear un diseño de página para documentos PDF destinados a una presentación en pantalla, ya sea a través de Internet o en salas con videoproyección. Está inspirado en el paquete *pdfscreen* y tiene unas prestaciones similares, aunque su desarrollo es diferente. En general, debemos usarlo con la clase *article*, aunque es posible utilizarlo con cualquier clase que defina los comandos *\section*, *\subsection* y *\subsubsection*. El paquete depende fuertemente del paquete *hyperref*, y requiere la versión 6.56 o posterior de éste. Utilizar el paquete *web* con versiones anteriores del paquete *hyperref* puede producir resultados inesperados.

Entre las opciones que admite el paquete hay que incluir necesariamente el «controlador», que puede ser uno de los que aparecen en el cuadro 7.1. En el preámbulo de nuestro documento no es necesario incluir explícitamente el paquete *hyperref* con sus opciones, ya que *web* se encarga de incluirlo con las opciones adecuadas, así como los paquetes *color* y *amssymb*. No obstante, es posible después de cargar el paquete *web* utilizar el comando *\hypersetup* para personalizar algunas opciones del paquete *hyperref* (véase la apartado I.8.3). En función de las opciones seleccionadas, el paquete también incluirá el paquete *eso-pic* (que, a su vez, llama al paquete *everyshi*).

El paquete soporta numerosas opciones, para propósitos muy diversos, que iremos explicando en los apartados siguientes. Como suele ser habitual, el paquete procesa, en primer lugar, el archivo de configuración *web.cfg*, en caso de que éste exista. En dicho archivo se pueden incluir nuestras opciones por defecto, que pueden ser modificadas para cada documento utilizando el argumento opcional cuando cargamos el paquete.

7.1.1. Idiomas

El paquete admite buena parte de los idiomas soportados por el paquete *babel*: *spanish*, *french*, *german*, *norsk*, *dutch*, *italian*, *russian*, *dansk*, *polish* y *finnish*. Conviene destacar, no obstante, que los textos predefinidos en los diferentes idiomas están definidos por el paquete y son independien-

tes de los comandos del paquete babel. Para personalizar dichos textos, en caso de que no nos gusten los predefinidos por el paquete, debemos modificar los siguientes comandos (los valores mostrados son los usados por defecto en el paquete):

```
\renewcommand\web@versionlabel{Versión}
\renewcommand\web@toc{Tabla de Contenido}
\renewcommand\web@continued{cont.}
\renewcommand\web@article{Inicio\hyperlink{\web@Start.1}{Artículo}}
\renewcommand\web@directory{Directorio}
\renewcommand\web@revision{Actualizado el:}
\renewcommand\web@copyright{Copyright}
\renewcommand\web@section{Sección}
\renewcommand\web@back{Volver}
\renewcommand\web@doc{Doc}}
```

Observemos que estos comandos contienen el carácter reservado @, por lo que si los utilizamos en nuestro documento deben situarse entre los comandos \makeatletter y \makeatother. Esta precaución no es necesaria si las modificaciones se incluyen en el archivo de configuración web.cfg. A la vista de los comandos anteriores, un buen candidato a ser modificado es \web@toc; una definición más adecuada sería la siguiente:

```
\renewcommand\web@toc{Índice general}
```

7.1.2. Visualización del documento PDF

Como el objetivo del paquete es producir documentos PDF que vayan a ser visualizados en el computador o ser proyectados con un videoproyector, el diseño de página es muy diferente de los diseños existentes en las clases de documento estándar. El paquete tiene predefinidos distintos modelos de dimensiones de la pantalla a través de las siguientes opciones (los valores que aparecen son *Ancho*×*Alto*):

- designi: 4,67 in×3,72 in (aprox. 11,89 cm×9,47 cm)
- designii: 5 in×4,5 in (aprox. 12,73 cm×11,45 cm)
- designiii: 6 in×5 in (aprox. 15,27 cm×12,73 cm)
- designiv: 5 in×4 in (aprox. 12,73 cm×10,18 cm)
- designv: 6 in×4,5 in (aprox. 15,27 cm×11,45 cm)

En todos los casos, los márgenes están fijados en 0,25 in, excepto el margen superior que vale 24 pt. Si ninguna de estas opciones se adapta perfectamente a nuestras necesidades, entonces no debemos cargar ninguna de tales opciones y utilizar en su lugar los siguientes comandos:

```
\screensize{Altura}{Anchura}
\margins{Izquierdo}{Derecho}{Superior}{Inferior}
```

El primero de ellos fija el tamaño de la pantalla y los cuatro argumentos del segundo especifican las distancias que separarán el texto de los lados izquierdo, derecho, superior e inferior de la página. Usualmente las presentaciones se visualizan a pantalla completa, de modo que para un mismo

monitor ocurre que si incrementamos las dimensiones de la zona destinada al texto, entonces la sensación visual es que disminuye el tamaño de los caracteres (y al revés).

En principio no existen restricciones en las dimensiones, por lo que se puede elegir cualquier valor y cualquier unidad, aunque no nos podemos olvidar del tamaño de la pantalla, pues éste determinará cuáles serán las dimensiones óptimas. Por razones obvias, dichos comandos deben ir en el preámbulo de nuestro documento y sólo pueden ser utilizados una vez.

7.1.3. Página del título

Para obtener una buena presentación es indispensable que la primera página sea especial, tanto por su diseño como por la información que contenga. Para construir la página del título, el paquete proporciona los siguientes comandos:

<code>\author{Autor}</code>	<code>\subject{Asunto}</code>	<code>\university{Universidad}</code>
<code>\title{Título}</code>	<code>\keywords{PalabrasClave}</code>	<code>\email{DirecciónElectrónica}</code> <code>\version{Versión}</code>

Estos siete comandos pueden utilizarse en el preámbulo para construir la página del título con el comando `\maketitle`. La información proporcionada por los cuatro comandos de las dos primeras columnas también se escribe, en su caso, en la ficha de datos del documento PDF. En este sentido, debemos recordar que la información para la ficha de datos debe seguir las especificaciones del lenguaje PDF, por lo que debemos utilizar los comandos `\pdfstringdef` o `\texorpdfstring`, descritos en la página 409. Así, el título del documento podría incluirse como sigue:

```
\title{La función \texorpdfstring{$e^x$}{exp(x)}}
```

El paquete implementa el comando

```
\optionalpagematter
```

que inicialmente tiene un valor vacío, para permitir incorporar manualmente información adicional que será utilizada por el comando `\maketitle`. Por tanto, para servir a su propósito, la redefinición del comando `\optionalpagematter` debe ir antes que el comando `\maketitle`. Por defecto, dicha información aparece después del autor y antes del «directorio» (véase el apartado siguiente).

El aspecto de la página del título está controlado además por los cuatro parámetros siguientes (entre paréntesis figura el valor por defecto):

`\titleauthorproportion` (0,33) Proporción vertical de la altura del texto (`\textheight`) que utilizarán el *Título* y el *Autor*.

`\hportionwebtitle` (0,7) Proporción horizontal de la longitud de línea (`\ linewidth`) que ocupará el *Título*.

`\hportionwebauthor` (0,4) Proporción horizontal de la longitud de línea (`\ linewidth`) que ocupará el *Autor*.

`\minimumskip` (`\medskip`) Mínima separación vertical entre los elementos principales de la página de título.

EJEMPLO 7.1

```
\title{Experimentos con el paquete web}
\author{CLMPS}\subject{Test del paquete web}
\keywords{LaTeX, hyperref, PDF, web}
\university{Universidad de Murcia\\
    Departamento de Matemáticas}
\email{latex@um.es}\version{1.0}
\renewcommand\optionalpagematter{\par
    \minimumskip\vspace{\stretch{1}}
    \begin{center}\fcolorbox{blue}{webyellow}{%
        \begin{minipage}{.67\linewidth}%
            \noindent\textcolor{red}{\textbf{Nota:}} Este
            texto ha sido obtenido redefiniendo el comando
            \texttt{\textbackslash optionalpagematter} para
            ilustrar su funcionamiento.
        \end{minipage}\end{center}}
\end{minipage}\end{center}}
```

Universidad de Murcia
Departamento de Matemáticas

Experimentos con el paquete web

CLMPS

Nota: Este texto ha sido obtenido redefiniendo el comando `\optionalpagematter` para ilustrar su funcionamiento.

Directorio

- Tabla de Contenido
- [Inicio Artículo](#)

Copyright © 2003 latex@um.es
Actualizado el: 29 de Abril de 2003

Versión 1.0



Otra alternativa interesante es redefinir el comando `\maketitle` para adaptarlo a nuestra necesidades. En este caso, y para aprovechar toda la información introducida por los comandos anteriores, debemos tener en cuenta que el paquete define, con dicha información, los siguientes comandos:

<code>\webauthor</code>	<code>\websubject</code>	<code>\webuniversity</code>
<code>\webtitle</code>	<code>\webkeywords</code>	<code>\webemail</code> <code>\webversion</code>

Entre otras acciones, el comando `\author{Autor}` define el comando `\webauthor` igual a la cadena *Autor* (y análogamente el resto de los comandos). Naturalmente, también es posible construir manualmente la página del título sin necesidad de utilizar el comando `\maketitle`.

7.1.4. El «directorío» y el índice general

El paquete redefine el comando `\maketitle` para crear su propio diseño de la página del título, incorporando a la misma el «directorío» del documento que, en particular, incorpora un enlace al índice general. Dos opciones controlan el comportamiento del paquete en este aspecto:

`nodirectory` Si el documento que estamos escribiendo es corto, o bien consiste en una lista (de ejercicios o de descripciones, por ejemplo), quizás no sea una buena idea incluir un índice general. En este caso, no incluiremos el comando `\tableofcontents` al comienzo del documento y entonces es preferible que no aparezca el «directorío». Para ello debemos utilizar esta opción, que se ocupa de suprimir el «directorío» de la página del título.

`latextoc` Para documentos largos, sin embargo, es muy conveniente incluir un índice general con enlaces a las diferentes secciones del documento, que nos ayude a navegar cómodamente por el mismo. El paquete ha rediseñado el índice general, suprimiendo el número de las páginas, numerando sólo las secciones y subsecciones, y colocando todas las unidades que son inferiores en la jerarquía unas a continuación de otras (véase la figura 7.1). Esta opción

Tabla de Contenido

- 1. Opciones**
 - 1.1. Controladores**
 - pdftex • dvips • dvipsone • dviwindo • textures
 - 1.2. Visualización del documento**
 - Opciones • Comandos asociados
- 2. Página de título**
 - 2.1. Comandos**
 - author • title • subject • keywords • university • email • version
 - 2.2. Parámetros**
 - titleauthorproportion • hproportionwebtitle • hproportionwebauthor • minimumskip
- 3. Navegación**
 - 3.1. Barra de navegación**
 - Activación • Configuración
 - 3.2. Panel de navegación**

Figura 7.1: Índice general en el paquete web

nos permite recuperar el diseño clásico de L^AT_EX; caso de utilizarla, puede ser conveniente activar también la opción `linktocpage` en el paquete `hyperref`.

7.1.5. Filigranas y fondos

El paquete puede utilizar un gráfico que sirva como filigrana o marca al agua, es decir, que sirva de fondo a la página. Para ello el paquete debe cargarse con la opción `usetemplates`. El comando para insertar una filigrana es el siguiente:

```
\template{ArchivoGráfico}
```

donde *ArchivoGráfico* hace referencia al nombre del fichero gráfico que deseamos incorporar y que debe tener una de las extensiones reconocidas por el «controlador» que utilicemos.

EJEMPLO 7.2

```
\usepackage[pdftex,designi,spanish,%
           usetemplates]{web}
...
\sfamily
\template{overlay1.pdf}
\section{Donald E. Knuth}
Uno de los objetivos que Donald E. Knuth se marcó cuando, allá por los años setenta, desarrolló el proyecto TEX (y que constituye una de sus señas de identidad) fue que los ficheros .dvi generados debían ser independientes de la plataforma, en el sentido de que pudieran ser utilizados con idénticos resultados en todas las plataformas salvo, naturalmente, la resolución con que apareciesen definidos los textos.

\section{HTML}
El lenguaje HTML, creado también para ser independiente de la plataforma, está diseñado para que los documentos sean leídos en la pantalla de un ordenador y no en papel. Su característica esencial es la interactividad, de tal forma que mediante "enlaces" podemos viajar a otras páginas en el mismo documento o en otros documentos, que pueden incluso estar situados en máquinas remotas accesibles vía internet.
```

Sección 1: Donald E. Knuth

3

1. Donald E. Knuth

Uno de los objetivos que Donald E. Knuth se marcó cuando, allá por los años setenta, desarrolló el proyecto T_EX (y que constituye una de sus señas de identidad) fue que los ficheros .dvi generados debían ser independientes de la plataforma, en el sentido de que pudieran ser utilizados con idénticos resultados en todas las plataformas salvo, naturalmente, la resolución con que apareciesen definidos los textos.

2. HTML

El lenguaje HTML, creado también para ser independiente de la plataforma, está diseñado para que los documentos sean leídos en la pantalla de un ordenador y no en papel. Su característica esencial es la interactividad, de tal forma que mediante "enlaces" podemos viajar a otras páginas en el mismo documento o en otros documentos, que pueden incluso estar situados en máquinas remotas accesibles vía internet.

En lugar de una imagen podemos especificar un color para el fondo con el comando

```
\textBgColor{NombreColor}
```

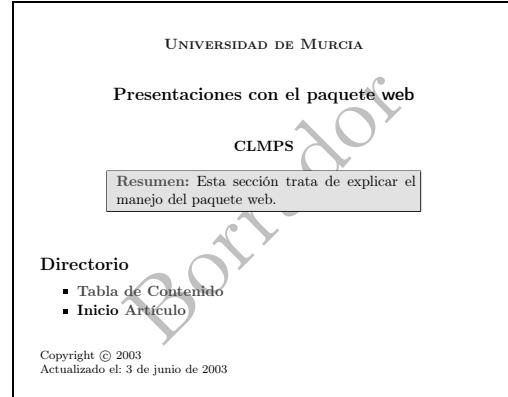
donde *NombreColor* indica el nombre de uno de los colores existentes (o que previamente hayamos definido, véase la sección 3.1). El paquete también dispone de un comando de propósito más general:

```
\AddToTemplate{NombreComando}
```

mediante el cual es posible incluir todo tipo de objetos en el fondo de la región destinada al texto. Hay que tener en cuenta que los objetos se colocarán en la esquina inferior izquierda de las páginas. Como ilustración véase el ejemplo 7.3, donde colocamos la palabra «Borrador» como marca al agua en el documento. Obsérvese que en el argumento del comando `\AddToTemplate` aparece el nombre del comando pero sin la antibarra \.

EJEMPLO 7.3

```
\usepackage[pdftex,designi,spanish,%
           usetemplates]{web}
\title{Presentaciones con el paquete \textsf{web}}
\author{CLMPS}
\university{\textsc{Universidad de Murcia}}
\renewcommand\optionalpagematter{\vfill
\begin{center}\fcolorbox{blue}{yellow}{%
\begin{minipage}[.67\linewidth]{%
\noindent\textcolor{red}{\textbf{Resumen:}} Esta sección trata de explicar ...
\end{minipage}}\end{center}}
\def\Borrador{\parbox[b]{\paperheight}{%
\textwidth\textscreenwidth\hfill
\rotatebox{45}{\color{webgray}
\scalebox{6}{Borrador}}\hfill\null\vfill}}
\AddToTemplate{Borrador}
```



Utilización del comando `\AddToTemplate`. La longitud `\textscreenwidth` es una longitud interna del paquete que determina la anchura máxima del texto

Los tres comandos anteriores afectan a la parte principal del texto, pero no siempre al panel de navegación (véase la sección 7.1.7), cuya alteración requiere de otros específicos. Para modificar la filigrana del panel disponemos del siguiente comando:

```
\paneltemplate{ArchivoGráfico}
```

Para modificar el color del panel de navegación puede utilizarse el comando

```
\panelBgColor{NombreColor}
```

Con el mismo propósito que el comando `\AddToTemplate` analizado anteriormente, pero referido ahora al fondo del panel de navegación, está el comando

```
\AddToPanelTemplate{NombreComando}
```

mediante el cual es posible incluir todo tipo de objetos en el fondo de la zona de pantalla destinada al panel de navegación. Cada uno de estos comandos puede utilizarse cuantas veces se desee y su

efecto comenzará a partir del momento en el que se use. Las filigranas y colores de fondo pueden ser desactivados mediante los siguientes comandos:

\ClearAllTemplates	\ClearTextTemplate	\ClearPanelTemplate
--------------------	--------------------	---------------------

7.1.6. Barra de navegación

Cuando preparamos un documento PDF para realizar una presentación, suele ser habitual que visualicemos el documento a pantalla completa, por ejemplo, incluyendo el comando

\hypersetup{pdfpagemode=FullScreen}

Sin embargo, esta opción lleva aparejada la desventaja de que entonces no son visibles ni la «barra de herramientas» ni la «barra de menús» del visor, por lo que la navegación a través del documento se hace más rígida e incómoda. Para remediar esta dificultad, el paquete dispone de la opción `navibar`, que crea una pequeña barra de navegación en la parte inferior del documento que apenas ocupa espacio, lo cual resulta muy práctico (véase el ejemplo 7.4).

El paquete utiliza un estilo de página propio, `webheadings`, que funciona como el estilo `headings` salvo que el pie de página, en lugar de estar vacío, contiene la barra de navegación (si la opción `navibar` ha sido seleccionada). En la cabecera de las páginas aparece el título y número de la sección (\rightmark) junto con el número de página. Si se desea eliminar el primero, lo cual puede ser adecuado para una presentación, basta redefinir `\rightmark` como vacío, como se hace en el siguiente ejemplo.

EJEMPLO 7.4

```
\usepackage[pdftex,designi,spanish,navibar]{web}
\title{Presentaciones con el paquete \textsf{web}}
\author{CLMPS}
\university{\textsc{Universidad de Murcia}}
\begin{document}
\def\righmark{}
\section{Presentaciones con el paquete
\textsf{web}}
Esta sección describe el uso del paquete
\textsf{web}, desarrollado por D.P. Story, que
está concebido para ser utilizado cuando la salida
es \textsc{pdf}. No obstante, las capacidades
completas de este paquete sólo se pueden
visualizar si el documento final es ...
\end{document}
```

3

1. Presentaciones con el paquete web

Esta sección describe el uso del paquete `web`, desarrollado por D.P. Story, que está concebido para ser utilizado cuando la salida es PDF. No obstante, las capacidades completas de este paquete sólo se pueden visualizar si el documento final PDF es visualizado con el programa ACROBAT READER.

El propósito del paquete `web` es crear un diseño de página para documentos PDF destinados a una presentación en pantalla, ya sea a través de Internet o en salas con videoproyección. Está inspirado en el paquete `pdfscreen` y tiene unas prestaciones similares, aunque su desarrollo es diferente. En general, debemos usarlo con la clase `article`, aunque es posible utilizarlo con cualquier clase que defina los comandos `\section`, `\subsection` y `\subsubsection`. El paquete depende fuertemente del paquete `hyperref`, y requiere una versión igual o superior a la versión 6.56. Utilizar el paquete `web` con versiones anteriores del paquete `hyperref` puede producir resultados inesperados.

Entre las opciones que admite el paquete, necesariamente hay que incluir el «controlador», que puede ser uno de los siguientes:

Toc ▶◀ ▶▶ ◀▶ Volver ◀ Doc Doc ▶

Barra de navegación del paquete `web` cuando se ha seleccionado la opción `navibar`

La barra de navegación puede ser activada y desactivada usando los siguientes comandos:

\NaviBarOff	\NaviBarOn
-------------	------------

que pueden utilizarse tantas veces como sea necesario. En el estilo predeterminado de la barra de navegación los recuadros son grises y los enlaces aparecen en color azul. Para cambiar estos colores debemos utilizar los comandos

\navibarBgColor{*Color1*}

\navibarTextColor{*Color2*}

que fijan, respectivamente, los colores del fondo y del texto de la barra de navegación. Los colores *Color1* y *Color2* deben ser nombres de colores reconocidos por el paquete color.

Si la barra de navegación proporcionada por el paquete no satisface sus necesidades, no cargue la opción navibar y construya una barra de navegación personalizada utilizando el comando

\insNaviBar{*Contenido*}

donde el argumento *Contenido* es totalmente libre. Por ejemplo, podríamos construir la siguiente barra de navegación, cuyo resultado se muestra en la figura 7.2:

```
\insNaviBar{\footnotesize\hfill\hyperlink{webtoc}{\$\\subsepeq\$}
    \Acrobatmenu{FirstPage}{%
        \$\\blacktriangleleft\\hspace{-3pt}\\blacktriangleleft\$}
    \Acrobatmenu{PrevPage}{\$\\blacktriangleleft\$}
    \Acrobatmenu{GoToPage}{?}
    \Acrobatmenu{NextPage}{\$\\blacktriangleright\$}
    \Acrobatmenu{LastPage}{%
        \$\\blacktriangleright\\hspace{-3pt}\\blacktriangleright\$}\\quad
        \Acrobatmenu{GoBack}{\$\\curvearrowleft\$}
        \Acrobatmenu{GoBackDoc}{\$\\vartriangleleft\$}
        \Acrobatmenu{GoForwardDoc}{\$\\vartriangleright\$}
        \Acrobatmenu{FullScreen}{\\raisebox{-1pt}{\$\\boxplus\$}}
        \Acrobatmenu{Close}{\$\\otimes\$}
        \Acrobatmenu{Quit}{\$\\bigodot\$}}%
```

El comando \insNaviBar puede ser desactivado, en cualquier momento, con el comando

\insNaviBarOff

Sin embargo, después de utilizar este comando no se recupera la barra de navegación estándar del paquete, sino que a partir de entonces el documento carece de barra de navegación.

7.1.7. Panel de navegación

Si deseamos utilizar un panel de navegación, existen las siguientes opciones, que pueden utilizarse conjuntamente con la opción *usetemplates*:

leftpanel | rightpanel El panel se coloca en la parte izquierda o derecha de la pantalla.

El diseño del panel de navegación se deja enteramente a la imaginación del usuario, que puede crear un panel para cada uno de sus documentos. El comando que se encarga de ello es

\buildpanel{*ObjetosConFormato*}

Sección 1: Presentaciones con el paquete web 3

1. Presentaciones con el paquete web

Esta sección describe el uso del paquete `web`, desarrollado por D.P. Story, que está concebido para ser utilizado cuando la salida es PDF. No obstante, las capacidades completas de este paquete sólo se pueden visualizar si el documento final PDF es visualizado con el programa ACROBAT READER.

El propósito del paquete `web` es crear un diseño de página para documentos PDF destinados a una presentación en pantalla, ya sea a través de Internet o en salas con videoproyección. Está inspirado en el paquete `pdfscreen` y tiene unas prestaciones similares, aunque su desarrollo es diferente. En general, debemos usarlo con la clase `article`, aunque es posible utilizarlo con cualquier clase que defina los comandos `\section`, `\subsection` y `\subsubsection`. El paquete depende fuertemente del paquete `hyperref`, y requiere una versión igual o superior a la versión 6.56. Utilizar el paquete `web` con versiones anteriores del paquete `hyperref` puede producir resultados inesperados.

Entre las opciones que admite el paquete, necesariamente hay que incluir el «controlador», que puede ser uno de los siguientes:

Figura 7.2: Barra de navegación personalizada en el paquete `web`

donde el argumento *ObjetosConFormato* es cualquier colección de elementos de L^AT_EX que se componen dentro de un entorno `center`. El panel se construye entonces en una caja con dimensiones

`\panelwidth{Anchura} \paperheight`

La *Anchura* asignada a `\panelwidth` ha de ser mayor o igual que 1in, pues en caso contrario es redefinida internamente a 1in. Por defecto, el argumento *ObjetosConFormato* es vacío, lo que explica, por ejemplo, que la opción `rightpanel` no haga nada. Cuando realmente existe un panel (ya sea a la izquierda o a la derecha), es decir, cuando *ObjetosConFormato* no es vacío, el comando

`\panelssep{Separación}`

especifica la separación entre el panel y texto principal. Después de utilizar `\panelssep` debemos ejecutar también el comando

`\InitLayout`

para que el paquete reconstruya la página teniendo en cuenta las nuevas longitudes. Conviene indicar que los comandos anteriores pueden utilizarse más de una vez, con objeto de poder cambiar el panel a lo largo del documento.

Cuando estamos diseñando el panel podemos emplear el comando

`\panelNaviGroup`

que permite incluir todos los botones de navegación. Estos botones de navegación pueden ser adecuadamente redefinidos para crear nuevos botones personalizados y adaptados a nuestras necesidades. Pero antes de poner un ejemplo de personalización de este grupo de navegación señalemos que el autor del paquete ha olvidado hacer las traducciones a los diferentes idiomas del botón ‘Close’ en `\panelNaviGroup`. El código que sigue realiza esta tarea para el idioma español:

\@newNaviIconMenu{\panel@Close}{28pt}{15pt}{\footnotesize Cerrar}{Close} y, en su caso, debería ser incluido en el preámbulo del documento (entre \makeatletter y \makeatother).

EJEMPLO 7.5

```
\usepackage[pdftex,designi,spanish,rightpanel]{web}
\buildpanel{\href{http://www.matematicas.um.es}
{\includegraphics[width=.95in]{umu.png}}
\par\vfill\href{http://www.latex.um.es/}
{\rotatebox{-90}{\parbox{1.6in}
{\Huge\color{red}\textsf{LaTeX}}\\
\normalsize\smash{\raisebox{5pt}{\color{blue}
\slshape\kern10pt para\kern18pt todos}}}}%
\par\vfill\panelNaviGroup }
\begin{document}
\section{Presentaciones con el paquete
\textsf{web}}
Esta sección describe el uso del paquete
\textsf{web}, desarrollado por D.P. Story, ...
\end{document}
```

Sección 1: Presentaciones con el paquete web 3

1. Presentaciones con el paquete web

Esta sección describe el uso del paquete web, desarrollado por D.P. Story, que está concebido para ser utilizado cuando la salida es PDF. No obstante, las capacidades completas de este paquete sólo se pueden visualizar si el documento final PDF es visualizado con el programa ACROBAT READER.

El propósito del paquete web es crear un diseño de página para documentos PDF destinados a una presentación en pantalla, ya sea a través de Internet o en salas con videoproyección. Está inspirado en el paquete pdfscreen y tiene unas prestaciones similares, aunque su desarrollo es diferente. En general, debemos usarlo con la clase article, aunque es posible utilizarlo con cualquier clase que defina los comandos \section, \subsection y \subsubsection. El paquete depende fuertemente del paquete hyperref, y requiere una versión igual o superior a la versión 6.56. Utilizar el paquete web con



Para construir nuevos botones de navegación, con propiedades diferentes, podemos utilizar el comando

```
\newNaviIcon{Tipo}{NombreComando}{Anchura}{Altura}{Texto}{Acción}
```

donde

Tipo puede ser uno de los siguientes caracteres: m, j, l, y sirve para identificar el tipo del nuevo botón de navegación. El tipo m utiliza el comando \Acrobatmenu para construir un botón de enlace. El tipo j se emplea para acciones arbitrarias que se realizan a través de código JavaScript. Y el tipo l se usa para construir enlaces a direcciones existentes.

NombreComando es el comando que identifica al nuevo botón de navegación.

Anchura es una longitud que determina la anchura del nuevo botón de navegación.

Altura es una longitud que determina la altura del nuevo botón de navegación.

Acción es la acción que se ejecutará al activar el botón, y dependerá del tipo del botón. En el tipo m puede ser una de las acciones Named (véase el cuadro 6.4 en la pág. 412). En el tipo j debe ser un ‘script’ escrito en código JavaScript. Y en el tipo l debe ser un comando \hyperlink o \href con su primer argumento.

No sólo los botones de navegación pueden ser redefinidos, sino que también el propio comando \panelNaviGroup, que los agrupa, puede ser modificado. Una posible redefinición sería la siguiente, que es semejante a la efectuada para la barra de navegación en la página 441:

```
\renewcommand\panelNaviGroup{\parbox[c]{58pt}{%
\Acrobatmenu{FirstPage}{%
\$blacktriangleleft\hspace{-3pt}\blacktriangleleft\$}\hfill
\Acrobatmenu{PrevPage}{\$blacktriangleleft\$}\hfill}}
```

```
\Acrobatmenu{NextPage}\{$\blacktriangleright$\}\hfill
\Acrobatmenu{LastPage}\{%
    $\blacktriangleright\hspace{-3pt}\blacktriangleright$\}\par
\Acrobatmenu{GoToPage}\{$\mathbb{P}$\}\hfill
\Acrobatmenu{GoBack}\{$\curvearrowleft$\}\hfill
\Acrobatmenu{FullScreen}\{$\square$\}\hfill
\Acrobatmenu{Close}\{$\otimes$\}\par
\centerline{\Acrobatmenu{Quit}\{$\bigodot$\}}}
```

El comando

`\ClearBuildPanel`

elimina el panel creado con el comando `\buildpanel`.

7.1.8. Otros comandos y opciones

Para finalizar el estudio del paquete web presentamos algunas opciones y comandos adicionales, de propósitos diversos.

`forpaper`, `forcolorpaper` Opciones que deben activarse cuando la salida no es la pantalla, sino que deseamos imprimir el documento en una impresora monocromo o color, respectivamente.

`tight` Realiza algunas definiciones de tamaños y longitudes para que el texto aparezca más compacto. Especialmente indicada cuando la salida es la pantalla del computador.

`\NewPage` Uno de los problemas del diseño de página para pantalla es que la altura del texto es bastante menor que la altura normal cuando imprimimos en papel. Por otra parte, cuando realizamos una presentación puede no ser muy importante que algunas páginas sean más cortas que otras: lo importante es que una determinada información aparezca en la misma página. Este comando permite romper página cuando la salida es la pantalla del computador (opción por defecto), pero deja de funcionar si se ha optado por una salida impresa (es decir, si se ha seleccionado alguna de las dos opciones analizadas en el primer ítem de esta lista).

7.2. Una herramienta adicional para las presentaciones: PPOWER4

PARA procesar documentos PDF que han sido creados con PDFLATEX, VLATEX o DVIPDFM, y conseguir que partes de una página sean mostradas paso a paso durante una presentación con ACROBAT READER, podemos utilizar PPOWER4, [26]. Entre las propiedades más interesantes de esta utilidad destacan las siguientes:

- Permite mostrar diferentes partes de una página en cualquier orden.
- Puede eliminar ciertas partes de una página.
- Permite seleccionar diversos efectos de transición entre las diferentes partes de una página.
- Admite degradados de colores como fondo de las páginas.

Esta utilidad no produce el documento PDF original, el cual debe ser creado con PDFLATEX, VLATEX o DVIPDFM junto con algún paquete específico para presentaciones (por ejemplo, web). El

objetivo principal de PPOWER4 es añadir efectos especiales al documento PDF creado. El proceso habitual de trabajo con PPOWER4 es el siguiente:

1. Con ayuda de un paquete diseñado especialmente para crear documentos PDF que vayan a ser visualizados en el computador, o ser proyectados con un videoproyector, creamos un documento PDF, por ejemplo, MiDoc.pdf, cuidando que cada página muestre la información adecuada.
2. A continuación incluimos los paquetes que vienen con PPOWER4 en el documento fuente, incorporamos al mismo los comandos necesarios para producir los «efectos» deseados, y volvemos a generar de nuevo el archivo MiDoc.pdf.
3. Finalmente procesamos el documento MiDoc.pdf con la utilidad ppower4.bat, que necesita dos argumentos: el fichero de entrada y el de salida. Por ejemplo:

```
ppower4 midoc.pdf midoc.pp4.pdf
```

De esta forma, el nuevo documento midoc.pp4.pdf será exactamente igual a MiDoc.pdf, pero con los efectos que hayamos incluido (pausas, fondos, etc.).

¿Qué necesito?

Para poder ejecutar PPOWER4 es necesario tener instalado Java 2 y haber descargado en el computador las librerías que vienen con PPOWER4, de forma que Java pueda localizarlas. La aplicación para MS-WINDOWS proporciona un archivo ppower4.bat que se encarga de procesar el archivo PDF. La ruta que figura en dicho archivo debe ser congruente con su instalación de Java. En una instalación por defecto, dicho archivo debería ser, más o menos, el siguiente:

```
@ECHO OFF
REM You must define path for pp4p.jar and name of the java command
REM Here the settings for the Sun Java Runtime Environment on my machine
set basedir="C:\Archivos de programa\JavaSoft\jre\1.3\lib"
set javacommand="C:\Archivos de programa\JavaSoft\jre\1.3\bin\java"
%javacommand% -cp %basedir%\pp4p.jar de.tu_darmstadt.sp.pp4.PPower4 %1 %2 %3 %4 %5
```

7.2.1. Los paquetes

PPOWER4 viene acompañado de los siguientes ficheros de estilo para L^AT_EX: pause, background y mpmulti, que iremos describiendo a lo largo de esta sección. Si trabajamos con PDFTEX o VLATEX se cargan sin opciones de controlador:

```
\usepackage{pause,background}
```

mientras que si trabajamos con DVIPDFM, entonces es necesario incluir la opción dvipdfm:

```
\usepackage[dvipdfm]{pause,background}
```

Una opción importante de los paquetes anteriores es ignore, que permite desactivar todos los comandos de pause y background sin necesidad de modificar el documento fuente. De esta manera podemos fácilmente transformar un documento que utiliza el postproceso de PPOWER4 en otro documento que no utiliza ninguna de las posibilidades de este programa.

7.2.2. Pausas: el paquete `pause`

Para construir páginas de manera incremental, sólo es necesario incluir el comando

`\pause`

en aquellas posiciones de la página donde queramos que la presentación haga una pausa. El comando hay que colocarlo justo al lado de donde queremos realizar la pausa, y esto será señalizado con un pequeño rectángulo de color, que desaparecerá después de ser procesado con PPOWER4.

Considere el documento del ejemplo 7.6. Si lo compilamos se producirá la página correspondiente (observe los rectángulos). El documento tiene una página, pero tras procesarlo con PPOWER4, obtenemos un nuevo documento PDF, esta vez de cinco páginas (véase la figura 7.3).

EJEMPLO 7.6

```
\documentclass[11pt,landscape]{article}
...
\usepackage{pause}
\title{Un primer ejemplo con PPower4}
\author{CLMPS}
\date{Abril de 2001}
\begin{document}
\begin{itemize}
\item Portugal\pause
\item España\pause
\begin{itemize}
\item Madrid\pause
\item Barcelona\pause
\end{itemize}
\item Francia
\end{itemize}
\end{document}
```

Documento PDF antes de ser procesado con PPOWER4

Un primer ejemplo con PPower4

CLMPS

Julio de 2003

- Portugal
- España
 - Madrid
 - Barcelona
- Francia

Si no deseamos que aparezca el pequeño rectángulo, entonces debemos tomar la precaución de cargar el paquete `pause` con la opción `nomarkers`:

```
\usepackage[nomarkers]{pause}
```

7.2.3. Efectos de transición

Ya sabemos que ACROBAT permite varios efectos de transición cuando cambia de una página a otra, aunque realmente ésta es una propiedad de cada página. En un documento PDF procesado con PPOWER4 existen dos tipos de transiciones: entre páginas originales y entre páginas virtuales (o subpáginas) creadas por PPOWER4.

Transiciones entre páginas originales

Podemos utilizar las herramientas que el paquete `hyperref` pone a nuestra disposición, o bien utilizar el archivo `pagetrans.tex` (de M. van Dongen) que viene con PPOWER4. Dicho archivo

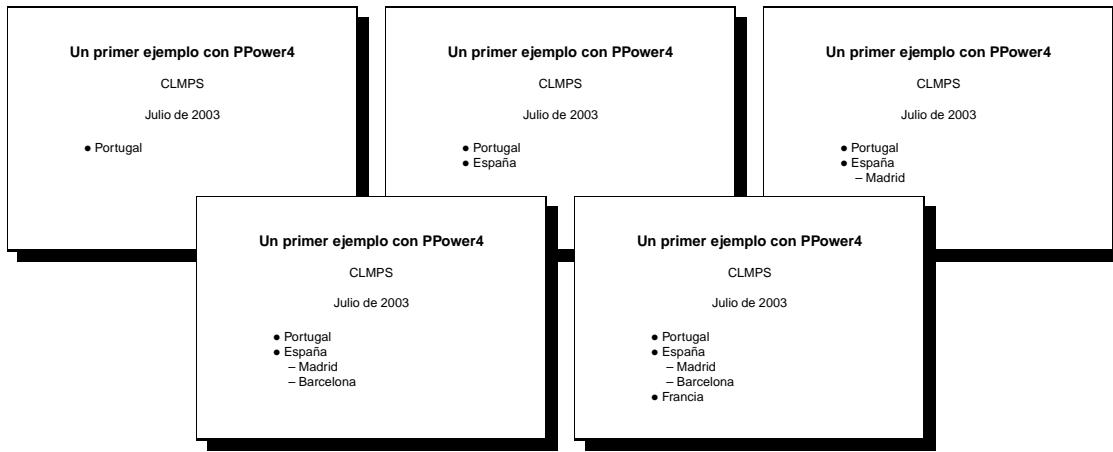


Figura 7.3: Sub-páginas del documento PDF del ejemplo 7.6 después de ser procesado con PPOWER4

habilita los siguientes comandos, cuya explicación resulta bastante obvia a partir de los diferentes efectos de transición (véase el cuadro 6.3):

\Replace	\Dissolve	\HBlinds
\VBlinds	\HOSplit	\HISplit
\VOSplit	\VISplit	\OBox
\IBox	\Wipe{Ángulo}	\pageTransitionGlitter{Ángulo}

Transiciones entre páginas virtuales (o subpáginas)

Los documentos PDF sólo admiten una transición por cada página, de modo que si entre cada dos pausas (generadas con el comando \pause) incluimos un efecto de transición diferente, ACROBAT sólo utilizará uno para todas las páginas virtuales creadas. Para permitir diferentes efectos de transición entre páginas virtuales, el paquete pause habilita una serie de comandos, creados a partir de \pause, que permiten especificar el efecto de transición de cada una de las páginas virtuales creadas. Estos comandos son los siguientes (como antes, no se requiere explicación):

\pauseReplace	\pauseDissolve	\pauseHBlinds
\pauseVBlinds	\pauseHOSplit	\pauseHISplit
\pauseVOSplit	\pauseVISplit	\pauseOBox
\pauseIBox	\pauseWipe{Ángulo}	\pauseGlitter{Ángulo}

7.2.4. Fondos: el paquete background

Los documentos de las presentaciones suelen ir sobre fondos de color y con el texto en un color de contraste. La acertada elección de los colores es esencial para el éxito de la presentación, y depende de muchos factores (entre otros, de la luminosidad de la sala donde vaya a realizarse la presentación). El paquete background, que se distribuye junto con PPOWER4, permite especificar

un color para el fondo, así como una graduación de colores, tanto en sentido horizontal como vertical. ¡Los resultados pueden ser espectaculares! Los comandos encargados de ello son:

`\pagecolor{NombreColor}`

que especifica el color plano que servirá como fondo de las páginas y

`\hpagecolor[Color1]{Color2} \vpagecolor[Color1]{Color2}`

Con estos comandos, el color del fondo cambia gradualmente (en sentido horizontal o vertical, respectivamente) desde el *Color1* hasta el *Color2*. Si el argumento opcional *Color1* no está presente, entonces el cambio se realiza desde los tonos oscuros hasta los tonos claros de *Color2*.

En los tres comandos, los nombres de los colores utilizados deben existir previamente en L^AT_EX. Como es natural, una vez hemos especificado un color para el fondo, éste permanece activo hasta que se selecciona un color distinto.

Si sólo deseamos utilizar colores planos para el fondo de las páginas (es decir, si sólo vamos a utilizar el comando `\pagecolor`), entonces no es necesario cargar el paquete `background` ya que PDFL^AT_EX interpreta adecuadamente este comando sin necesidad de ningún paquete adicional.

Añadiendo otros elementos al fondo

Si quiere añadir otros elementos al fondo de las páginas (como texto o figuras), en primer lugar debe tener instalado el paquete `eso-pic`, que es automáticamente incluido por el paquete `background`. Cuando carguemos este paquete debemos instruirle para que pueda manejar y añadir otros elementos al fondo, y esto se hace con la opción `bgadd`:

`\usepackage[bgadd]{background}`

Con esta opción activada, el paquete proporciona dos nuevos comandos para situar los objetos en el fondo:

`\bgadd{Objeto}`

`\bgaddcenter{Objeto}`

El primero coloca el *Objeto* en la esquina superior izquierda de la página, mientras que el segundo lo coloca centrado en la página. Algunos ejemplos válidos son:

`\bgadd{\includegraphics[width=20mm]{logo.pdf}}`

`\bgaddcenter{\rotatebox{45}{\color{Gray}\scalebox{6}{Borrador}}}`

Todos los comandos `\bgadd` y `\bgaddcenter` van añadiendo los objetos correspondientes al fondo actual de las páginas, es decir, tienen un comportamiento acumulativo. El comando

`\bgclear`

permite eliminar todos los elementos que forman el fondo de las páginas.

7.2.5. Enlaces con las primeras subpáginas

Cuando incluimos un hiperenlace a una determinada página en un documento PDF, ésta siempre hace referencia a la página completa o, en otras palabras, a la última subpágina o porción,

si hemos utilizado el comando `\pause`. Esto puede resultar adecuado si retrocedemos en la presentación para mostrar una transparencia que ya ha sido visualizada por la audiencia. Pero, ¿y si queremos avanzar a una página que todavía no ha sido visitada? Entonces seguramente no desearemos mostrar la nueva página completa, sino sólo la primera subpágina, para no romper así la sorpresa que debe acompañar a toda presentación.

Para poder enlazar con la primera subpágina de una página necesitamos el paquete `pp4link`, que viene en la distribución de PPOWER4. Este paquete introduce dos nuevos comandos:

`\toptarget{Nombre}`

`\toplink{Nombre}{Objeto}`

que se utilizan para construir los enlaces a la primera subpágina de las páginas que se construyen de manera incremental. La sintaxis de estos dos comandos coincide con la de los comandos `\hypertarget` y `\hyperlink` que proporciona el paquete `hyperref`. Para que los enlaces funcionen correctamente es aconsejable compilar el documento PDF dos veces. El *Nombre* del enlace sólo puede estar formado por letras, no estando permitidos números ni ningún otro carácter. Si en el documento existen varias páginas con el mismo número, entonces estos enlaces pueden no funcionar correctamente y, después del postproceso, estos enlaces pueden conectar con cualquiera de las páginas con la misma numeración.

Debemos señalar que con la actual versión de PPOWER4 no es posible construir enlaces a subpáginas arbitrarias; de momento sólo se puede enlazar con la primera subpágina (utilizando `\toptarget` y `\toplink`) y con la última subpágina, que ya es la página completa (mediante `\hypertarget` y `\hyperlink`).

7.2.6. Los entresijos de PPOWER4

En el apartado anterior hemos visto que el comando `\pause` divide cada página en porciones que son mostradas poco a poco, hasta que la página se completa. Pero el mecanismo descrito no permite visualizar las distintas porciones en un orden distinto del secuencial, que está determinado por el orden en que aparecen en el documento fuente. La construcción de las páginas con las porciones respectivas, pero en un orden distinto, no es tarea sencilla.

Imaginemos que cada porción de una página es numerada (1, 2, 3...) conforme aparece en el documento fuente. El proceso ordinario consiste en visualizar las porciones en ese mismo orden (1, 2, 3...). Por defecto, la primera porción a ser visualizada es la número 1, y cada comando `\pause` incrementa en uno el contador que se ocupa de determinar la porción que será también visualizada en la siguiente página virtual. Veámoslo con un ejemplo.

Supongamos una página en la que hemos incluido tres comandos `\pause`:

... éste es el texto anterior a la primera pausa.`\pause`

Ahora estaríamos en la segunda porción (texto entre la primera y segunda pausas).`\pause` Llegamos a la tercera porción (texto entre la segunda y tercera pausas).`\pause` Y finalmente está el texto después de la última pausa (la cuarta porción)

En la figura 7.4 se muestran las cuatro páginas virtuales que los comandos `\pause` crean. Como vemos, la porción 1 se muestra en las páginas virtuales 1, 2, 3 y 4; la porción 2 se muestra en las

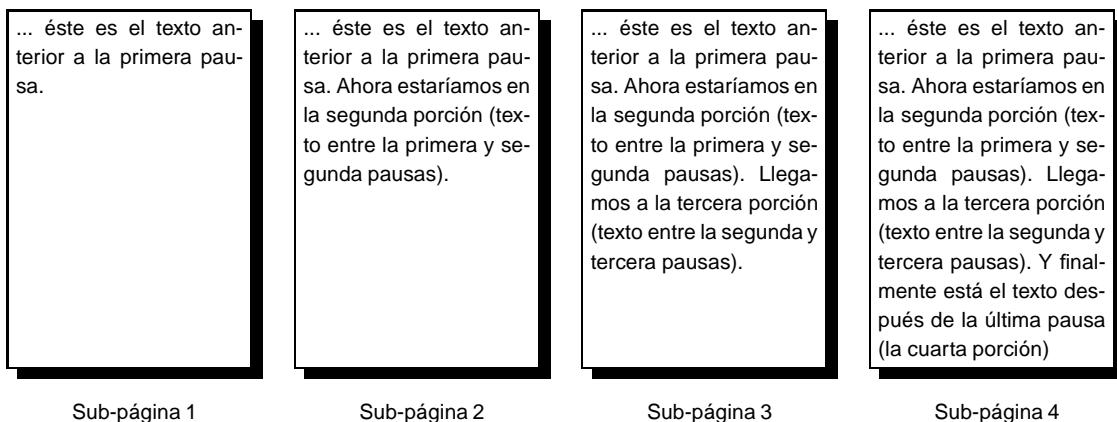


Figura 7.4: Páginas virtuales creadas por PPOWER4

páginas virtuales 2, 3 y 4; la porción 3 se muestra en las páginas virtuales 3 y 4; la porción 4 sólo se muestra en la página virtual 4 (que es la página completa). Por tanto, si asignamos a cada porción un nivel (igual, en este caso, a su número de orden), una porción se visualizará en una página virtual sólo si su nivel no es superior al nivel actual de la página virtual. Con este criterio general, si modificamos manualmente el nivel de una porción podemos hacer que aparezca en cualquier página virtual. Veamos cómo se hace esto.

7.2.7. Asignando niveles a las porciones

Para la asignación de niveles, el paquete pause proporciona el comando

`\pauselevel{Niveles1, Niveles2...}`

que permite asignar a la porción donde se encuentra dicho comando los niveles *Niveles1*, *Niveles2*, etc. Dentro de cada uno de los *Niveles* existen diferentes caracteres de control. Son los siguientes:

- = Asigna el nivel de manera absoluta.
- =+ Asigna el nivel aumentando el nivel actual.
- Asigna el nivel disminuyendo el nivel actual.
- :
- :+ Especifica el nivel máximo absoluto a partir del cual la porción no se visualizará.
- :- Especifica el nivel máximo relativo a partir del cual la porción no se visualizará.
- :- Especifica el nivel mínimo relativo a partir del cual la porción no se visualizará. Tiene sentido cuando los comandos \pause van disminuyendo el nivel.

Ilustremos con algunos ejemplos estos comandos:

- `\pauselevel{=7}` asigna el nivel 7 a la actual porción.
- `\pauselevel{=+4}` asigna a la actual porción un nivel igual al actual aumentado en 4. Así, si el nivel actual es 7, el comando anterior asigna el nivel 11 a la porción actual.
- `\pauselevel{:8}` especifica que la porción se mostrará en todas las páginas virtuales cuyo nivel esté en el rango 1 a 8.

- `\pauselevel{=3 :5}` especifica que la porción será visible en las páginas virtuales cuyo nivel sea 3, 4 o 5 (obsérvese el espacio en blanco que hay antes de :).
- `\pauselevel{=3 :5, =8 :10}` indica que la porción será mostrada en los niveles 3 a 5 y en los niveles 8 a 10.

Para facilitar la asignación de niveles, el comando permite especificar el valor (positivo o negativo) en que se modificará el nivel con cada comando `\pause`. Por defecto, al comienzo de cada página el valor de incremento es +1, pero es posible modificar esto. Por ejemplo, `\pauselevel{=11 -1}` asigna el nivel 11 a la porción y especifica que los siguientes comandos `\pause` irán reduciendo el nivel en una unidad. Obsérvese el espacio en blanco que hay entre 11 y -1.

7.2.8. Resaltando las porciones

Otra técnica en las presentaciones, muy interesante también, consiste en construir la página completa e ir coloreando posteriormente las distintas porciones. De este modo, todas las páginas virtuales contienen la misma información; lo que cambia son los diferentes colores utilizados. Para conseguir esto, el paquete proporciona el siguiente comando:

```
\pausecolors{ColorCambio}{ColorNormal}{ColorDestacado}
```

Dicho comando permite reemplazar un color por otro para conseguir el efecto deseado. Más precisamente, todo el texto de color *ColorCambio* se imprime en *ColorNormal* cuando está en una porción no activa, y en *ColorDestacado* cuando se encuentra en la porción activa; los elementos que estén en un color diferente de *ColorCambio* no sufrirán ninguna modificación. Todos los reemplazamientos se acumulan y son realizados en una única ocasión, lo que evita problemas a PPOWER4. Obviamente, los tres colores anteriores no tienen por qué ser distintos, aunque no tiene sentido utilizar el comando con los tres colores iguales. Imaginemos el siguiente documento de tres porciones:

```
\definecolor{gris}{gray}{0.4} \pausecolors{red}{gris}{magenta}
\begin{itemize}\color{red}
\item Éste es un punto importante.\pause
\item \textcolor{blue}{Pero sólo hasta que aparece el siguiente.}\pause
\item Y ambos dejan de ser importantes cuando aparece el tercero.
\end{itemize}
```

Si lo compilamos y procesamos con PPOWER4 obtendremos un documento de una página con tres subpáginas. El primer ítem será de color magenta en la primera subpágina y de color gris en las otras dos subpáginas; el segundo ítem será siempre de color blue; y el tercer ítem será de color gris en las dos primeras subpáginas y de color magenta en la última subpágina.

Dado que los comandos `\pausecolors` son acumulativos, es posible que en un determinado momento no sepamos cuál puede ser el resultado de todos juntos, por lo que sería deseable poder desactivarlos todos. Esto se consigue con el siguiente comando:

```
\pausicolorreset
```

7.2.9. Modos de trabajo

PPOWER4 puede interpretar los comandos \pause de dos maneras distintas: en modo resaltado (highlight), consistente en cambiar los colores de la porción activa, y en modo incremental (build), que muestra las porciones en el mismo orden en que aparecen en el documento fuente.

En el modo resaltado el número que indica el nivel no hace referencia al nivel en que una porción aparecerá, sino al nivel en que dicha porción será resaltada. Además, el nivel final, aquel a partir del cual una porción deja de ser visualizada en el modo incremental, ahora hace referencia al nivel a partir del cual la porción dejará de estar resaltada. Si no especificamos ninguna, la porción permanecerá resaltada hasta el final.

Para seleccionar un determinado modo debemos utilizar los siguientes comandos:

\pausebuild

\pausehighlight

que permiten activar el modo incremental (por defecto) o el modo resaltado, respectivamente.

Ambos modos de trabajo pueden combinarse perfectamente. Supongamos que estamos trabajando en modo incremental y queremos que una porción sea visible desde el comienzo de la página. Entonces podemos marcarla con la clave highlight cuando le asignamos los niveles. De este modo será resaltada cada vez que corresponda. La manera de hacerlo se ilustra en el siguiente ejemplo:

```
... \pause\pauselevel{highlight =2 :5}Este texto aparece
resaltado sólo en los niveles 2 a 5.\pause ...
```

Si sólo queremos que la porción aparezca resaltada cuando sea visualizada (de acuerdo con su nivel), entonces bastará marcar dicha porción sólo con la clave highlight.

Supongamos ahora que estamos trabajando en modo resaltado y ciertas porciones sólo deben visualizarse en algunos niveles. Entonces debemos marcar dichas porciones con la etiqueta build. Además, también podemos combinar los diferentes modos en una misma porción, como se muestra en el siguiente texto:

```
... \pause\pauselevel{build =2 :5, =3}Este texto sólo es visible
en los niveles 2 a 5, y será resaltado en el nivel 3.\pause ...
```

7.3. Presentaciones con la clase prosper

La clase prosper, diseñada por F. Goualard y P. M. Neergaard [24], permite crear presentaciones electrónicas de una excelente calidad. Con ella se pueden producir documentos en formato PDF para realizar exposiciones con un monitor o con un sistema de proyección de vídeo, un computador y el programa ACROBAT READER. También se puede producir un documento PostScript para imprimir la presentación sobre transparencias para exposiciones con retroproyector. Al utilizar esta clase se cargan los paquetes seminar, PSTricks, graphicx, hyperref, amssymb y times. Para poder utilizar la clase debemos estar seguros de tenerlos instalados en nuestro sistema.

La estructura de un documento de esta clase es muy simple; en la figura 7.3 está representada de forma esquemática. Un documento fuente para esta clase debe contener en el preámbulo las

```
\documentclass[Opciones]{prosper}

\title{Título de la presentación}
\subtitle{Subtítulo}
\author{Autores}
\institution{Institución o Empresa}
\slideCaption{Anotaciones y Leyenda}

\begin{document}
\maketitle

\begin{slide}[Transición]{Título de la pantalla}
    Contenido de la pantalla
\end{slide}
...
\overlays{Número de animaciones}{%
\begin{slide}[Transición]{Título de la pantalla}
    Contenido de la pantalla
\end{slide}
}
...
\end{document}
```

Figura 7.5: Estructura de un documento \LaTeX para la clase **prosper**; los comandos sobre gris son optativos

declaraciones del título y el autor de la exposición, y el cuerpo contendrá tantos entornos **slide** como páginas (transparencias o pantallas) se deseé incluir en la presentación.

7.3.1. Estilos de presentación

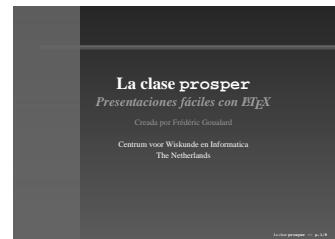
La elección del estilo es el primer paso cuando nos disponemos a crear una presentación con **prosper**, porque determina las dimensiones del texto en cada transparencia, el gráfico del fondo y los colores de las letras.

La declaración del estilo se realiza incluyendo el nombre del estilo como opción de la clase. En cualquier momento podemos cambiar de estilo, pero siempre con la precaución de revisar la presentación ante los posibles cambios de color y posición del contenido de las pantallas.

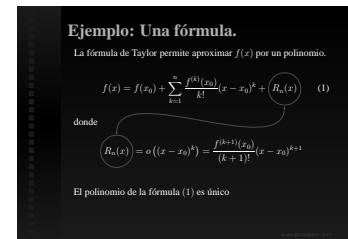
En la figura 7.6 se muestran ejemplos de los diferentes estilos relacionados en el cuadro 7.2. En este cuadro están todos los estilos creados por el autor de la clase, contenidos en la distribución de la misma, junto con los estilos **gyom**, **pascal** y **rico** que son contribuciones adicionales de diferentes autores (es posible crear nuevos estilos siguiendo las instrucciones incluidas en la documentación [24]). Desde el sitio de Internet de **prosper** se pueden descargar estilos adicionales.



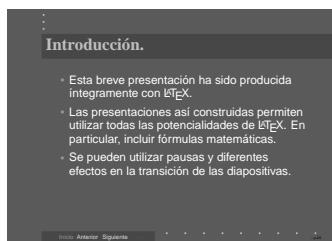
alien



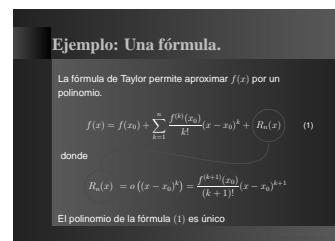
autumn



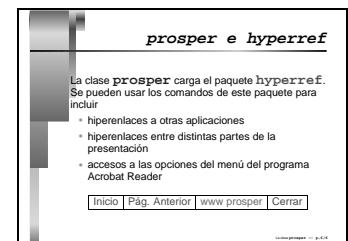
azure



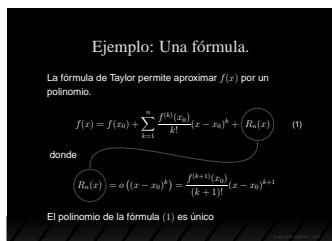
contemporain



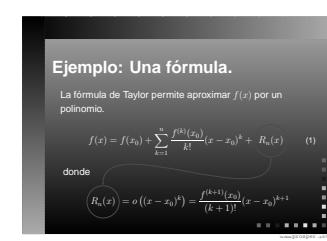
darkblue



frames



lignesbleues



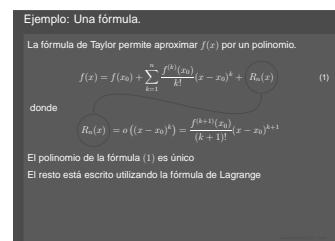
nuancegris



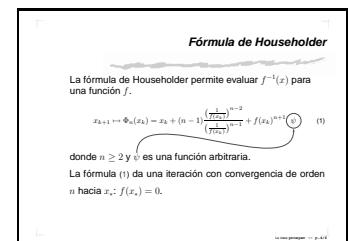
troispoints



gyom



pascal



rico

Figura 7.6: Estilos de la clase prosper

alienglow	autumn	azure	contemporain
darkblue	frames	lignesbleues	nuancegris
troispoints	gyom	pascal	rico

Cuadro 7.2: Estilos de las presentaciones con *prosper*

7.3.2. Opciones de la clase *prosper*

Además del estilo de la presentación, las opciones disponibles de esta clase son las siguientes (en negrita aparecen las opciones que carga por defecto).

- ps | pdf** La compilación del documento con la opción **ps** produce un fichero PostScript para imprimir transparencias para utilizar un retroproyector clásico. Con la opción **pdf** se produce un fichero en formato PDF con la presentación electrónica para ser expuesta en una pantalla o con un sistema de proyección de vídeo. En los dos casos las dimensiones de las páginas coinciden, los cambios se producen en la selección de colores para el fondo y las letras, que son diferentes en función del sistema de exposición elegido.
- draft | final** Con la opción **draft** la leyenda del pie de cada página incluye el nombre del documento fuente, el título, el autor de la presentación y la fecha de la compilación. La opción **final** incluye en la leyenda de las páginas el contenido del comando `\slideCaption` (v. pág. 456) si éste se ha utilizado, o el título de la presentación en caso contrario. Además, esta opción es trasladada al paquete `graphicx` (véase la sección 4.2.1).
- total | nototal** Si se usa la opción **total** en la leyenda aparece el número de la página actual junto con el número total de páginas. Con **nototal** sólo aparece el número de la página en curso.
- slideBW | slideColor** La opción **slideBW** restringe el uso de colores; está pensada para imprimir transparencias utilizando impresoras en blanco y negro. Por el contrario, **slideColor** permite utilizar todos los colores, por lo que debe usarse con precaución si se intenta hacer impresiones en blanco y negro.
- colorBG | nocolorBG** Con la opción **nocolorBG** el color del fondo de las páginas es siempre blanco, independientemente del estilo elegido. Esta opción es una buena elección si pretendemos imprimir la presentación en una impresora en blanco y negro. Con **colorBG** el color del fondo dependerá del estilo elegido.
- noaccumulate | accumulate** Cuando se crean presentaciones PDF es posible crear animaciones que se obtienen al superponer páginas. Con **noaccumulate** se permiten estas animaciones, mientras que con **accumulate** todas las páginas de la animación se acumulan en una sola página. Esta opción es interesante si se quiere tener una versión impresa de la presentación.

7.3.3. El proceso de compilación

Antes de entrar a detallar los elementos de un documento de la clase *prosper* es conveniente describir los procesos que nos van a proporcionar la presentación. Como la clase carga el paquete `PSTricks` para utilizar los recursos del lenguaje PostScript en la construcción de las transparencias,

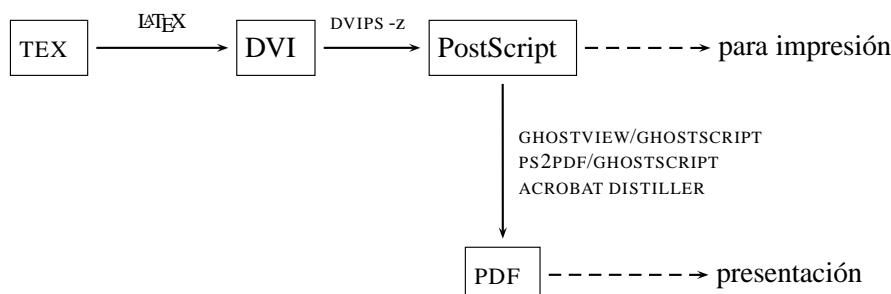


Figura 7.7: Construcción de una presentación con prosper

no es posible obtener la presentación en formato PDF compilando con PDFLATEX, ni haciendo actuar DVIPDFM sobre la salida DVI de LATEX.

El procedimiento para obtener las presentaciones aparece esquematizado en la figura 7.7. Se comienza creando un archivo PostScript al compilar el documento fuente con L^AT_EX y procesar la salida DVI con DVIPS (la opción `-z` es necesaria para construir los hiperenlaces de la salida PDF). Con estos dos pasos ya se puede imprimir la presentación PostScript. Para obtener el formato PDF es necesario convertir el documento PostScript obtenido siguiendo cualquiera de las estrategias de conversión descritas en la sección 4.1.1.

7.3.4. El preámbulo de un documento prosper

En el preámbulo de un documento escrito con la clase `prosper` es obligatorio incluir los siguientes comandos:

\title{*Título*} \author{*Autores*} que declaran el *Título* y los *Autores* de la presentación. Además de estos comandos, la clase proporciona los siguientes:

\subtitle{Subtítulo}	\email{E-mails}
\institution{Organismos}	\slideCaption{Leyenda}
\Logo(x,y){miLogo}	\Logo{miLogo}
\DefaultTransition{Transición}	\displayVersion

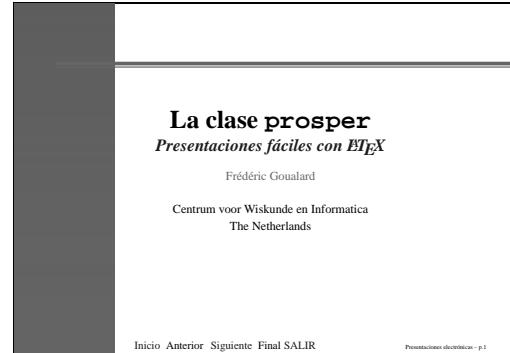
El comando \email almacena las direcciones de correo electrónico de los autores de la presentación. El argumento del comando \institution contiene los *Organismos* en los que trabajan los autores. \Logo permite incluir el gráfico *miLogo.eps* en todas las pantallas de la presentación en la posición (x,y) relativa a la esquina inferior izquierda de la zona de escritura, o en la posición predeterminada por el estilo en uso cuando no se dan las coordenadas (x,y) .

Con el comando \slideCaption se declara la *Leyenda* que aparecerá al pie de todas las pantallas (nombre de la conferencia, evento, etc.). También puede usarse para incluir una barra de navegación creada con los comandos de acceso al menú de ACROBAT READER, definidos en el paquete hyperref (véanse el apartado I.8.3 y el ejemplo 7.7). Si no se usa este comando se utilizará el *Título* de la presentación como contenido por defecto de la *Leyenda* de todos los pies de página.

EJEMPLO 7.7

```
\documentclass[ps,autumn,nototal]{prosper}
\usepackage[latin1]{inputenc}
\title{La clase \texttt{prosper}}
\subtitle{Presentaciones fáciles con \LaTeX{}}
\author{Frédéric Goualard}
\institution{Centrum voor Wiskunde en Informatica\\
The Netherlands}
\slideCaption{\small
\Acrobatmenu{FirstPage}{\color{blue}Inicio}\ \
\Acrobatmenu{PrevPage}{\color{black}Anterior}\ \
\Acrobatmenu{NextPage}{\color{blue}Siguiente}\ \
\Acrobatmenu{LastPage}{\color{black}Final}\ \
\Acrobatmenu{Quit}{\color{blue}SALIR}\hfill}
Presentaciones electrónicas
\begin{document} \maketitle \end{document}
```

Preámbulo de una presentación **prosper**



El comando `\DefaultTransition{Transición}` define el efecto de transición entre páginas de la presentación PDF. *Transición* es uno de los valores que aparecen en el cuadro 7.3 que especifica el modo de paso entre las páginas (véase también el cuadro 6.3). Los efectos de transición también se pueden declarar de manera independiente para cada cambio de pantalla, tal y como veremos en los apartados siguientes.

Blinds	Box	Dissolve
Glitter	Replace	Split
		Wipe

Cuadro 7.3: Efectos de transición entre páginas con **prosper**

La declaración `\displayVersion` hace que con la opción final aparezca en las páginas la misma leyenda que se usa en la opción `draft`, es decir, el título de la presentación, el autor y la fecha de la compilación.

7.3.5. Cuerpo de un documento **prosper**

El cuerpo del documento fuente de una presentación **prosper** puede comenzar (habitualmente será así) con el comando `\maketitle`, que producirá una primera página que contiene las declaraciones del preámbulo (título, autor, etc.). A continuación se va describiendo cada una de las páginas que van a formar parte de la presentación.

El contenido de cada una de estas páginas se introduce con un entorno `slide`:

```
\begin{slide}[Transición]{Título}
Contenido de la página
\end{slide}
```

Este entorno admite como argumento optativo el efecto de *Transición* que se quiere tener al acceder a esta página en la presentación PDF (véase el cuadro 7.3) y tiene como argumento obligatorio el *Título* de la página.

El contenido del entorno *slide* se representa en una sola página; es decir, si sobrepasa el tamaño de la página sólo queda visible una parte del contenido. Con la clase *prosper* el autor del documento debe decidir el contenido de cada una de las páginas.

Si nuestra presentación está dividida en distintas secciones o apartados con su correspondiente título, podemos incluir una página en el título de cada sección con el comando

```
\part [Transición] {TítuloDeLaSección}
```

que incluirá una nueva página, utilizando el efecto de *Transición* declarado, con el *TítuloDeLaSección* situado en el centro de la pantalla y escrito con la misma fuente que el título de la presentación.

Cada estilo de presentación tiene prefijadas las fuentes y los colores que se usarán en la presentación. Si se quiere modificar esta selección de manera global, o en una parte de la presentación, tanto en el preámbulo como en el cuerpo de un documento *prosper* se pueden utilizar las siguientes declaraciones, teniendo siempre en cuenta que afectan al documento desde el momento en el que aparecen y hasta que una nueva declaración no modifique su actuación.

<pre>\FontTitle{FuenteColor}{FuenteBW} \ColorFoot{Color}</pre>	<pre>\FontText{FuenteColor}{FuenteBW}</pre>
--	---

Los comandos *\FontTitle* y *\FontText* declaran, respectivamente, las fuentes y los colores a utilizar con el título y el contenido de las páginas. Cada comando necesita dos argumentos con las declaraciones de fuentes similares a las opciones de compilación: *slideColor* para tener presentaciones en color o *slideBW* para tener presentaciones que se imprimirán en blanco y negro.

El comando *\ColorFoot{Color}* declara el *Color* a utilizar en las leyendas de todas las páginas a partir de la actual.

<pre>\fontTitle{Texto}</pre>	<pre>\fontText{Texto}</pre>
------------------------------	-----------------------------

Los comandos *\fontTitle* y *\fontText* escriben su argumento, *Texto*, utilizando las fuentes del título o del contenido, respectivamente.

EJEMPLO 7.8

```
\documentclass[ps,contemporain]{prosper}
\begin{document}
\begin{slide}[Box]{Introducción.}
\begin{itemize}
\renewcommand{\labelitemi}{%
\includegraphics[width=.4cm]{%
green-bullet-on-white.ps}}
\item Esta breve presentación ha sido ...
\item Las presentaciones así construidas ...
\item Se pueden utilizar pausas y diferentes ...
\end{itemize}
\end{slide}
\end{document}
```

Una lista en *prosper*

En la clase `prosper` se ha redefinido el entorno `itemize` para las listas, haciendo que no justifique por la derecha, con lo que se evita que se produzcan divisiones de palabras (véase el ejemplo 7.8). Si se desea, es posible utilizar el entorno original de L^AT_EX, que está renombrado como `Itemize`.

La distribución de la clase incluye algunos ficheros gráficos que se pueden utilizar para modificar las viñetas de las listas. En el ejemplo 7.8 usamos el gráfico `green-bullet-on-white.ps` para nuestra lista, pero existen otros. Sus nombres siguen el esquema `color1-bullet-on-color2` donde `color1` puede ser `green`, `red` o `yellow`, y `color2` puede ser `blue` o `white`.

7.3.6. Animaciones en pantalla

En las presentaciones PDF construidas con `prosper` es posible conseguir efectos de animación en una pantalla haciendo que el contenido de la misma aparezca o desaparezca en distintas etapas (de forma incremental). Vamos a analizar los comandos que permiten conseguir este efecto, pero debemos tener presente que éstos sólo realizan su función con la opción `pdf`. Para imprimir en formato PostScript una presentación construida con animaciones necesitaremos hacer uso de la opción `accumulate` para acumular todas las etapas en una sola página.

Para hacer estas animaciones es necesario comenzar advirtiendo al compilador del número de etapas en las que se quiere mostrar la pantalla. Para ello, disponemos del comando `\overlays`, que debemos usar con la siguiente sintaxis:

```
\overlays{Número}{%
  \begin{slide}[Transición]{Título}
  ...
  \end{slide}}
```

Esta declaración tiene dos argumentos, el *Número* de etapas de la presentación de la página y la propia página definida en un entorno `slide`. Nótese que no puede haber espacios o líneas en blanco entre la línea donde está `\overlays` y `\begin{slide}`, ni tampoco entre el final de este entorno y la llave que encierra el argumento de `\overlays`.

También existe el entorno `itemstep` para construir listas en las que los ítems van apareciendo en pantalla de forma incremental. Su sintaxis coincide con la del entorno `item`.

```
\begin{itemstep}
\item Primer ítem
\item Segundo ítem
...
\end{itemstep}
```

En el ejemplo 7.9 podemos observar una primera aplicación del comando `\overlays`, actuando conjuntamente con el entorno `itemstep`.

Una vez declarado el *Número* de etapas de una animación con `\overlays`, dentro del entorno `slide` se pueden usar los siguientes comandos que controlan el texto que debe presentarse en cada uno de los pasos:

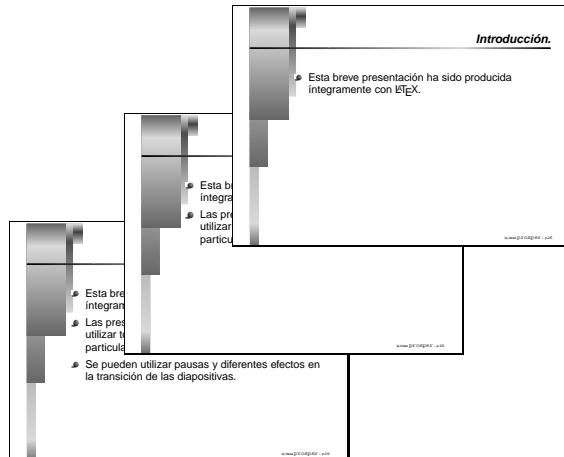
\fromSlide{p}{Texto}	\fromSlide*{p}{Texto}	\FromSlide{p}
\onlySlide{p}{Texto}	\onlySlide*{p}{Texto}	\OnlySlide{p}
\untilSlide{p}{Texto}	\untilSlide*{p}{Texto}	\UntilSlide{p}

El comando `\fromSlide` escribe el argumento *Texto*, desde la etapa *p* hasta el final de la animación; el comando `\FromSlide` escribe todo lo que sigue, desde la página *p* hasta el final de la animación. `\onlySlide` escribe su argumento *Texto* únicamente en el paso *p*, mientras que `\OnlySlide` escribe todo lo que sigue, sólo en la etapa *p*. Con `\untilSlide` se escribe su argumento *Texto*, desde el principio hasta la etapa *p* y `\UntilSlide` escribe todo lo que sigue, sólo hasta la etapa *p*.

EJEMPLO 7.9

```
\documentclass[pdf,frames]{prosper}
\begin{document}
\overlays{3}{%
\begin{slide}[Box]{Introducción.}
\begin{itemstep}
\item Esta breve presentación ha sido producida
integramente con \LaTeX.
\item Las presentaciones así construidas permiten
utilizar todas las potencialidades de \LaTeX.
En particular, incluir fórmulas matemáticas.
\item Se pueden utilizar pausas y diferentes
efectos en la transición de las diapositivas.
\end{itemstep}
\end{slide}
}

```



Animación de una lista. Representamos de arriba abajo las tres pantallas consecutivas que forman la animación de la lista

La forma en que el compilador \LaTeX construye las distintas etapas es simple: comienza construyendo las cajas de los distintos elementos definidos en el entorno `slide`; después, utilizando esas cajas, construye tantas páginas como indica el *Número* de `\overlays`, reemplazando las cajas que no deben aparecer en una etapa por cajas vacías pero con las mismas dimensiones, evitando así distorsiones en la posición de los elementos visibles.

Cuando queramos que al avanzar en las etapas algunos elementos sean sustituidos por otros, podemos utilizar las versiones de los comandos con asterisco *, que eliminan los elementos cuando no deben aparecer en una etapa.

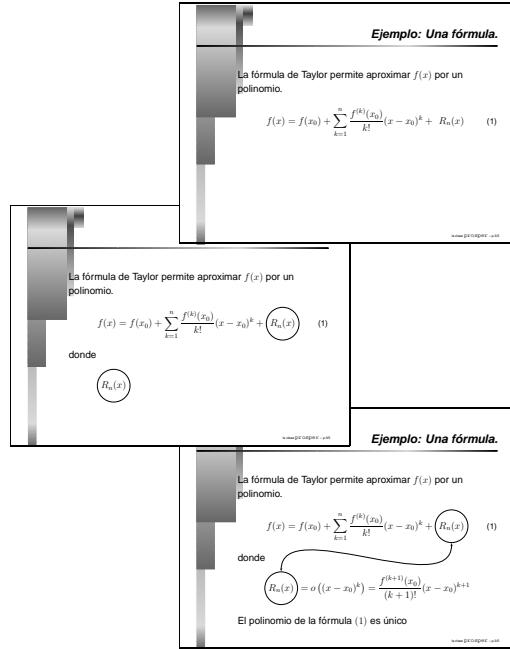
Obsérvese que para usar recursos gráficos de PStricks en distintas etapas es necesario utilizar los comandos con asterisco, porque `prosper` pasa las órdenes a \LaTeX , que sólo prepara el código PostScript de los gráficos PStricks, mientras que `DVIPS` es el encargado de crear y posicionar estos gráficos.

En el ejemplo 7.10 presentamos una animación construida con estos comandos en la que va apareciendo el texto en distintas etapas y que incluye una sustitución de la primera fórmula en la

segunda etapa, y algunos nodos y líneas de conexión entre ellos realizadas con `PSTricks` (véase la sección 4.4). En el ejemplo se han utilizado versiones con asterisco, con código `PSTricks`, en las tres etapas. También ha sido necesario incluir el primer nodo en color invisible para que no varíe en ninguna de las tres etapas el tamaño de la caja que contiene a la fórmula.

EJEMPLO 7.10

```
\overlays{3}{%
\begin{slide}{Ejemplo: Una fórmula.}
\small
La fórmula de Taylor permite aproximar
$f(x)$ por un polinomio.
\begin{equation}\label{Taylor}
f(x)=f(x_0)+\sum_{k=1}^n \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k+
\end{equation}
\onlySlide*{1}{%
\rnode{NA}{\pscirclebox[linestyle=none]{%
{R_n(x)}\nonumber\tag{1}}}}%
\fromSlide*{2}{%
\rnode{NA}{\pscirclebox{%
{R_n(x)}\nonumber\tag{1}}}}%
\end{equation}
\FromSlide{2}{%
donde $$\rnode{NB}{\pscirclebox{R_n(x)}}}%
\onlySlide*{3}{%
=o\left((x-x_0)^k\right)=
\frac{f^{(k+1)}(x_0)}{(k+1)!}(x-x_0)^{k+1}
}$$
\fromSlide*{3}{%
\nccurve[angleA=90,angleB=270]{<->}{NB}{NA}%
\OnlySlide{3}{%
El polinomio de la fórmula $(1)$ es único
}}%
\end{slide}
}
```



Representamos de arriba abajo las tres pantallas consecutivas que forman la animación de la página

PARA SABER MÁS

- En la página personal de D. P. Story podrá encontrar información adicional sobre «The AcroTeX eEducation Bundle». El conjunto está formado por cuatro paquetes: `web`, `exerquiz` (para crear ejercicios y cuestionarios on-line), `insdls` (para incluir código JavaScript en un documento `LATEX`) y `djslib` (que actúa como una librería de funciones JavaScript). La página web de «AcroTeX» es <http://math.uakron.edu/~dpstory/acrotex.html>.
- Las últimas versiones de `PPOWER4`, con ejemplos adicionales, se encuentran en la dirección <http://www-sp.iti.informatik.tu-darmstadt.de/software/ppower4/>.
- Las últimas versiones de la clase `prosper`, junto con más ejemplos y otras direcciones de Internet cuyo contenido está relacionado con `prosper`, pueden descargarse de la página <http://prosper.sourceforge.net/>.

Capítulo

8

Programación

NA de las principales ventajas de \TeX es su flexibilidad y versatilidad. Estas cualidades se suelen concretar, fundamentalmente, en que \TeX no es un programa cerrado con un número (aunque sea grande, ciertamente) de comandos y entornos predefinidos, sino que es programable, lo que permite a un usuario avanzado desarrollar sus propios comandos y entornos, adaptándolos completamente a sus necesidades. Este capítulo se centra más en \TeX que en \LaTeX , lo cual quiere decir que el tipo de comandos que presentaremos, y la utilización que de ellos vamos a hacer, seguramente no se ajustarán a las normas y criterios generales que hemos venido exponiendo en los capítulos precedentes. En primer lugar, presentaremos una nueva manera de definir comandos, diferente a las ya comentadas, pero que resulta extremadamente útil cuando nos movemos en el ámbito de la programación. A continuación comentaremos cómo construir y manipular los contadores y longitudes en \TeX , lo que permitirá definir contadores de forma local y realizar divisiones de contadores y longitudes, tareas que no era posible realizar utilizando la sintaxis propia de \LaTeX . También presentaremos algunos comandos propios de \TeX que tienen un comportamiento interesante, de suma utilidad en ciertas situaciones especiales. El capítulo finaliza analizando la programación de estructuras de control (los típicos condicionales y bucles característicos de cualquier lenguaje de programación de alto nivel), que aportan a \TeX un valor añadido del que carecen otras herramientas de procesamiento de textos.

8.1. Definiendo comandos de otro modo

E N la lección 11 ya hemos descrito cómo se pueden definir nuevos comandos en \LaTeX . La finalidad que se persigue cuando se define un nuevo comando es, entre otras, simplificar la escritura del documento y facilitar futuras revisiones del mismo. Por ejemplo, si estamos escribiendo un libro donde aparece muchas veces el símbolo \Rightarrow , una buena idea puede ser definir un comando `\fdl` (de «flecha doble larga»):

```
\newcommand*\fdl{\Longrightarrow}
```

Otra manera de definir nuevos comandos es la siguiente:

```
\def\NuevoComando{Definición}
```

El comando anterior podría haberse definido mediante `\def\f{d\{\Longrightarrow\}}`, que sólo puede utilizarse en modo matemático. Recordemos que para definir un comando que sirva en cualquier modo debemos utilizar `\ensuremath`; por ejemplo, el comando `\ecuacion` encargado de escribir $x_1^2 + x_2^2 + \dots + x_n^2 = 1$ se podría definir así:

```
\def\ecuacion{\ensuremath{x_1^2+x_2^2+\cdots+x_n^2=1}}
```

Al utilizar esta manera de construir comandos mediante `\def` debemos ser sumamente cuidadosos, ya que aunque el comando `\NuevoComando` ya existiese, no se produciría ningún error ni mensaje de advertencia, y puede darse el caso de que redefinamos comandos anteriores, pudiéndose generar así errores indeseados.

8.1.1. Los comandos son como funciones

Ya sabemos cómo construir comandos que admiten hasta nueve argumentos, de los cuales el primero puede ser opcional, de tal forma que si el usuario no proporciona dicho argumento, el comando trabajará con un valor por defecto. Para referirnos a los argumentos en la definición de un comando debemos utilizar el carácter almohadilla # seguido de un número (del 1 al 9) que hace referencia al argumento, según el orden que ocupa. Para ilustrar la definición de comandos con argumentos utilizando el comando primitivo `\def`, vamos a mejorar el comando `\ecuacion` considerado anteriormente, incluyendo como argumentos el nombre de la variable y el número de incógnitas n en la ecuación. Esto puede conseguirse con la siguiente definición:

```
\def\ecuacion#1#2{\ensuremath{#1_1^2+#1_2^2+\cdots+#1_{#2}^2=1}}
```

En este caso, cada ejecución del comando puede ser diferente, en función de sus argumentos. Ahora podríamos escribir `\ecuacion{y}{6}` para obtener $y_1^2 + y_2^2 + \dots + y_6^2 = 1$.

En general, la sintaxis que debemos utilizar para definir mediante `\def` un comando que depende de varios argumentos es la siguiente:

```
\def\NuevoComando#1..#9{Definición}
```

donde `\NuevoComando` es el nombre del nuevo comando, `Definición` es lo que debe hacer y `#1..#9` sirve para indicar los argumentos de los que depende.

8.1.2. Delimitadores de los argumentos

Cuando utilizamos comandos con argumentos, `TEX` debe conocer dónde empiezan y dónde acaban los argumentos. En general, esto se consigue colocando entre llaves cada uno de los argumentos, o entre corchetes los que son opcionales. En estos casos, las llaves o corchetes actúan como delimitadores de los argumentos. Sin embargo, una de las ventajas que posee definir los comandos utilizando `\def` es la posibilidad de elegir los delimitadores de los argumentos. En general, `TEX` lee los argumentos de manera secuencial: el primer carácter o grupo lo asigna al primer argumento, el segundo carácter o grupo lo asigna al segundo argumento, y así sucesivamente. Por ejemplo, si hacemos `\frac an`, el primer argumento #1 es `a` y el segundo argumento #2 es `n`; si hacemos `\frac b{n+1}`, la primera variable #1 es `b` y la segunda variable #2 es `n+1`; pero si hacemos `\frac{(A+B)}{n+1}`, la primera variable #1 es `(A+B)` y la segunda variable #2 es `n+1`.

Otra posibilidad interesante para indicar las variables consiste en utilizar delimitadores especiales entre los argumentos. Estos delimitadores pueden ser cualquier carácter o caracteres válidos, así como cualquier comando. Por ejemplo, si hacemos

```
\def\frac#1;#2:{\left(\frac{#1}{#2}\right)^2}
```

entonces al escribir `\frac{A+B};n+1`: estamos indicando que `(A+B)` es el primer argumento variable y `n+1` es el segundo. Los delimitadores son el punto y coma `<;>` para indicar que ha terminado el primer argumento y comienza el segundo argumento, y los dos puntos `<:>` para indicar que ha finalizado el segundo argumento. Un uso más sofisticado de los delimitadores se muestra en el siguiente ejemplo.

EJEMPLO 8.1

```
\def\nota #1;#2\linea{\par\noindent
\textbf{Nota #1:}\itshape #2 \upshape \par}
... delimitadores de los argumentos.
\nota 99;En general, \TeX{} lee los argumentos
de manera secuencial: el primer ... \linea
Un ejemplo más sofisticado ...
```

... delimitadores de los argumentos.

Nota 99: En general, *\TeX* lee los argumentos de manera secuencial: el primer carácter o caracteres ...

Un ejemplo más sofisticado ...

□ Cuando el texto de los argumentos puede ser cualquiera, debemos escoger un delimitador que no pueda aparecer como parte de uno de los argumentos. En estos casos, un comando no existente suele ser una buena idea; en este ejemplo hemos utilizado el comando `\linea`

8.1.3. Definiciones globales

El lugar donde hayamos definido un comando determinará el lugar donde podemos utilizarlo: si la definición está dentro de un grupo (véase la sección 10.1) su utilización estará limitada a ese grupo. Observemos en el ejemplo 8.2 cómo es posible modificar localmente la definición de un comando, sin afectar a su comportamiento global.

EJEMPLO 8.2

```
\def\MiComando{Definición global}\MiComando
\begin{center}
\def\MiComando{Definición centrada}\MiComando
\begin{list}{\textbullet}
\def\MiComando{Definición listada}\item\MiComando
\end{list}\MiComando
\end{center}\MiComando
```

Definición global

Definición centrada

- Definición lista

Definición centrada

Definición global

□ ¿Qué ocurre si se define un nuevo comando dentro de un grupo y es necesario utilizarlo fuera de él? Los comandos creados con `\newcommand*` o `\providecommand*` (o modificados mediante `\renewcommand*`) y `\def` son «locales», en el sentido de que su campo de acción es el grupo donde fueron definidos, lo cual posee ventajas indiscutibles. Sin embargo, a veces necesitamos eliminar esta restricción; para ello *\TeX* dispone de una forma especial de construir comandos, mediante la cual el comando amplía su campo de acción a todo lo que venga a continuación. Ya vimos en la lección 10 que los grupos tienen un punto de inicio, usualmente determinado por una

llave { o por el inicio de un entorno, y un punto final, que suele ser una llave } o el final del entorno que lo inició. Sin embargo, existen otras dos maneras de definir un grupo:

\bgroup	\begingroup
\egroup	\endgroup

donde los comandos de la primera columna son equivalentes, respectivamente, a las llaves { y }. Los comandos de ambas columnas no pueden combinarse para delimitar el grupo; es decir, cada grupo iniciado con un comando \bgroup (\begingroup, resp.) sólo puede ser cerrado con un comando \egroup (\endgroup, resp.). En algunos casos no es posible utilizar los delimitadores { y }; entonces resulta necesario emplear los comandos anteriores (véase la definición del entorno *textoculto* en el ejemplo 8.18).

Antes de presentar la forma en que T_EX permite ampliar el campo de acción de un comando, veamos primero una situación donde es necesario definir un comando de forma global. Supongamos que queremos crear un comando \comA, dependiente de un argumento, cuya misión es construir un segundo comando \comB cuyo valor sea el argumento del primer comando. El primer intento de definición sería el siguiente:

```
\def\comA#1{\def\comB{#1}}
```

Sin embargo, esta solución no funciona, ya que el comando \comB no puede utilizarse fuera del grupo en el que ha sido definido. De este modo, \comA{test}\comB produciría el conocido error «*Undefined control sequence*». La forma de solucionar este problema es declarando como global la definición del comando \comB, lo cual puede hacerse con cualquiera de las dos formas siguientes:

```
\def\comA#1{\global\def\comB{#1}}
\def\comA#1{\gdef\comB{#1}}
```

Para definir un comando global se puede utilizar cualquiera de los siguientes comandos:

\global\def\NuevoComando#1..#9{Definición}
\gdef\NuevoComando#1..#9{Definición}

Existen numerosos ejemplos en L^AT_EX que utilizan esta construcción. Por citar sólo algunos, en la clase article los comandos \title, \author y \date, dependientes de un argumento, redefinen globalmente los comandos \@title, \@author y \@date, respectivamente, los cuales son utilizados posteriormente por el comando \maketitle para generar la página del título. Algunas clases de documento utilizan estos datos para construir las cabeceras del artículo.

El carácter local también se traslada en T_EX, por ejemplo, a las asignaciones de los contadores. Sin embargo, L^AT_EX hace que las asignaciones de los contadores realizadas mediante los comandos \setcounter y \addtocounter sean globales (véase la lección 17). Para manipular los contadores de manera local debemos recurrir a los comandos primitivos de T_EX que se encargan de estas tareas (véase la sección 8.2). La asignación y modificación de longitudes (que se realiza con los comandos \setlength y \addtolength) son, sin embargo, locales, por lo que deben ir precedidas de la palabra clave \global si deseamos que su influencia no se limite al grupo en el que se han realizado.

8.1.4. Definiciones recursivas

La definición de un nuevo comando o la redefinición de uno ya existente no implica que se ejecuten los comandos que participan en la definición. Así, al definir `\def\aa{\bb}`, el comando `\aa` se almacena en memoria como `\bb`. T_EX no intenta ejecutar `\bb` en el momento de la definición; sólo cuando vamos a utilizar `\aa` es cuando T_EX ejecutará el comando `\bb` y todos los posibles comandos que aparezcan en la definición de `\bb`. Por ejemplo, al hacer `\def\salto{\vspace{1Km}}` T_EX no se para advirtiéndonos del error, ni siquiera protesta enviándonos una observación, pues Km no es una unidad de longitud admisible. Pero si intentamos ejecutar el comando `\salto`, entonces T_EX nos responderá con

```
! Illegal unit of measure (pt inserted).
<to be read again>
```

En ocasiones queremos que, al realizar la definición de un nuevo comando, se ejecuten los comandos que participan en su definición (lo cual no siempre es posible; en [32, págs. 212–215] se da una lista de todos los comandos que tienen dicha propiedad). Supongamos que hemos definido un comando `\aa`. ¿Es posible dar una nueva definición del comando `\aa` utilizando la definición anterior? Un primer intento sería el siguiente:

```
\def\aa{nada} \def\aa{Mucho más que \aa}\aa
```

Sin embargo, lo anterior provoca que T_EX no haga nada o bien que se quede «colgado» al entrar en un bucle infinito. En estos casos, lo que interesa es que cuando vamos a redefinir el comando `\aa` se sustituya antes `\aa` por su antiguo valor, es decir, que el comando `\aa` se ejecute o expanda. Esto se consigue definiendo el comando `\aa` mediante `\edef` en lugar de hacerlo de la manera habitual (mediante `\def` o `\newcommand`).

<code>\edef\NuevoComando#1..#9{Definición}</code>	<code>\xdef\NuevoComando#1..#9{Definición}</code>
---	---

El comando `\xdef` es equivalente a `\global\edef`. Por ejemplo, el código

```
\def\aa{nada} \edef\aa{Mucho más que \aa}\aa
```

produce: «Mucho más que nada». Observemos que se ha cambiado la segunda aparición de `\def` por `\edef`, para que el comando `\aa` se ejecute antes de redefinirlo.

Una manera simple de distinguir `\def` y `\edef` es la siguiente: los resultados de los comandos construidos mediante `\def` dependen de los valores de otros comandos en el momento de la ejecución, mientras que los resultados de los comandos construidos mediante `\edef` dependen de los valores de los otros comandos en el momento de la definición.

8.1.5. El comando `\let`

El comando `\let` sirve para definir comandos «constantes» a partir de la definición que en el momento de su construcción tiene asignada otro comando. La sintaxis general del comando es

<code>\let\NuevoComando=\ComandoExistente</code>
--

y se puede utilizar también con comandos que necesitan argumentos. En este caso, el nuevo comando constante también esperará el mismo número de argumentos.

En el siguiente ejemplo se muestra la diferencia existente entre `\def` y `\let` a la hora de construir nuevos comandos.

EJEMPLO 8.3

```

1 \def\uno{1} Uno: \uno \\
2 \let\UNO=\uno UNO: \UNO \\
3 \def\dos{\uno\uno} Dos: \dos\\
4 \def\ DOS{\UNO\UNO} DOS: \DOS\\
5 \def\uno{uno} Uno: \uno \\
6 Dos: \dos \\
7 DOS: \DOS
  
```

Los números de línea que figuran en la parte izquierda no deben escribirse, pues se han añadido al código fuente para referencias en el texto

En la línea 1 del Ejemplo 8.3 se define el comando `\uno` y en las líneas 2 a 4 se crean los comandos `\UNO`, `\dos` y `\DOS`. En la construcción de `\UNO` se sustituye el comando `\uno` por su definición, cosa que no ocurre en la construcción de `\dos`. Por esta razón, después de redefinir `\uno` en la línea 5, el comando `\dos` cambia, mientras que los comandos `\UNO` y `\DOS` no cambian.

Una asignación del tipo `\let\NuevoComando=\ComandoA\ComandoB` es incorrecta. Si queremos construir un comando `\NuevoComando` de la forma anterior, antes debemos crear un comando cuyo valor sea `\ComandoA\ComandoB`. Por ejemplo, imaginemos un comando `\pni` para crear un nuevo párrafo y escribir en negrita italizada. Una posibilidad para definirlo es la siguiente:

```
\def\temp{\par\bfseries\itshape} \let\pni=\temp
```

Con esto nos garantizamos que aunque posteriormente cambien las definiciones de alguno de los comandos que aparecen en la definición de `\temp`, el comando `\pni` no cambiará. De hecho ésta es la principal finalidad del comando `\let`: «sacar una copia» de un comando para que funcione siempre del mismo modo, sin verse afectado por futuros cambios en los comandos que utiliza.

8.2. Contadores y longitudes en \TeX

EN esta sección vamos a analizar muy brevemente cómo se manipulan los contadores y las longitudes con \TeX . Básicamente, \TeX puede realizar operaciones con tres clases de magnitudes: contadores (que corresponden a los registros tipo `count`), longitudes rígidas (que corresponden a los registros tipo `dimen`) y longitudes elásticas (que corresponden a los registros tipo `skip` y `muskip`, utilizadas estas últimas sólo en modo matemático). Pero, ¿cómo podemos definir estos registros? Existen cuatro comandos, según el tipo de registro:

<code>\newcount\NuevoConta</code>	<code>\newdimen\NuevaLong</code>
<code>\newskip\NuevaLong</code>	<code>\newmuskip\NuevaLong</code>

Tras estas asignaciones, el registro `\NuevoConta` sólo puede almacenar números enteros entre `-2147483647` y `2147483647`, ambos inclusive, mientras que el registro `\NuevaLong` puede almacenar cualquier longitud rígida (si se ha definido mediante `\newdimen`) o elástica (si se ha utilizado `\newskip` o `\newmuskip`). Antes de comentar los comandos de \TeX para la manipulación de los contadores y longitudes, conviene realizar las siguientes observaciones.

- Todas las longitudes creadas en \LaTeX utilizando el comando `\newlength` están definidas, con carácter local, por medio de `\newskip`. Concretamente \LaTeX da la siguiente definición:

$$\def\newlength#1{\@ifdefinable#1{\newskip#1}}$$

donde el comando `\@ifdefinable` se utiliza para garantizar que la definición es posible, ya que el número de registros que se pueden definir en \TeX está limitado a 256 por cada tipo o categoría.
- Todos los contadores creados en \LaTeX con el comando `\newcounter` se construyen internamente a partir de `\newcount`. Esencialmente, la instrucción `\newcounter{NuevoConta}` equivale a las siguientes líneas:

$$\newcount\c@NuevoConta$$

$$\setcounter{NuevoConta}{0}$$

$$\gdef\theNuevoConta{\arabic{NuevoConta}}$$

aunque la definición exacta es un poco más compleja.

Manipulación de contadores y longitudes

\TeX nos proporciona comandos para realizar las cuatro operaciones aritméticas básicas con los registros anteriores: suma, resta, multiplicación y división.

Para sumar (o restar) dos registros del mismo tipo disponemos del siguiente comando, en dos versiones equivalentes:

<code>\advance\NomRegistro \pmSum</code>	<code>\advance\NomRegistro by \pmSum</code>
---	--

donde *Sum* debe ser una longitud o un número entero del mismo tipo que `\NomRegistro`. El resultado de la operación se almacenará en `\NomRegistro`. Si sumamos (o restamos) dos longitudes elásticas cuyas componentes de «estiramiento» plus o «encogimiento» minus son infinitas, entonces las partes infinitas de orden inferior desaparecerán. Por ejemplo, después de los dos comandos siguientes

```
\LongElastica=10pt plus 1fil minus 2fill
\advance\LongElastica by 5pt plus 3fill minus 1fil
```

el valor de `\LongElastica` será `15pt plus 3fill minus 2fill`, pues de las dos componentes plus, 1fil y 3fill, esta última es la mayor, y de las dos componentes minus, 2fill y 1fil, la primera es de orden superior. Otros ejemplos se muestran a continuación.

EJEMPLO 8.4

```
\midim=5.5pt \miskip=10.5pt \micount=3
\midim=\the\midim, \miskip=\the\miskip,
\micount=\the\micount\ll[5mm]
\advance\midim 3pt \advance\miskip 4pt plus2pt
\advance\micount -5
\midim=\the\midim, \miskip=\the\miskip,
\micount=\the\micount
```

Valores iniciales:
`midim=5.5pt, miskip=10.5pt, micount=3`

Nuevos valores:
`midim=8.5pt, miskip=14.5pt plus 2.0pt, micount=-2`

En este ejemplo, y en los sucesivos de esta sección, supondremos que se han creado los siguientes registros:
`\newdimen\midim \newskip\miskip \newcount\micount`

Para multiplicar cualquier registro por un número disponemos del comando

<code>\multiply\NomRegistro Núm</code>	<code>\multiply\NomRegistro by Núm</code>
--	---

donde *Núm* es cualquier número, salvo que *\NomRegistro* sea un contador, en cuyo caso es un número entero. El resultado se almacena en *\NomRegistro*. Si se multiplica una longitud elástica, todas sus partes (tanto el valor «central» como los valores *plus* o *minus*) se ven afectadas, con la única excepción de las componentes *plus* o *minus* cuando éstas contienen valores infinitos.

EJEMPLO 8.5

```
\midim=5.5pt \miskip=10.5pt plus 3pt \micount=3
midim=\the\midim, miskip=\the\miskip,
micount=\the\micount\,[5mm]
\multiply\midim 3 \multiply\miskip 4
\multiply\micount -5
midim=\the\midim, miskip=\the\miskip,
micount=\the\micount
```

Valores iniciales:
`midim=5.5pt, miskip=10.5pt plus 3.0pt, micount=3`

Nuevos valores:
`midim=16.5pt, miskip=42.0pt plus 12.0pt, micount=-15`



Una forma alternativa de expresar la multiplicación de una longitud por un número es la siguiente, donde el signo igual de asignación puede omitirse:

<code>\Longitud1=Núm\Longitud2</code>

Las longitudes *\Longitud1* y *\Longitud2* pueden ser la misma. Por ejemplo, para hacer que la anchura del texto (que se almacena en la longitud *\textwidth*) sea el 90 % de la anchura actual escribiremos *\textwidth=0.9\textwidth*.

Finalmente, para dividir un registro (contador o longitud) por un número entero disponemos del comando siguiente:

<code>\divide\NomRegistro Núm</code>	<code>\divide\NomRegistro by Núm</code>
--------------------------------------	---

donde *Núm* es un número entero. El resultado se almacena en *\NomRegistro* y no siempre es la división exacta. En el caso de longitudes, éstas se transforman primero en unidades *sp* y el resultado se redondea a un múltiplo entero de esta unidad; en el caso de un registro tipo *count*, *\NomRegistro* almacenará la parte entera de la división.

EJEMPLO 8.6

```
\midim=5.5pt \miskip=10.5pt \micount=3
midim=\the\midim, miskip=\the\miskip,
micount=\the\micount\,[5mm]
\divide\midim 3 \divide\miskip 4
\divide\micount 5
midim=\the\midim, miskip=\the\miskip,
micount=\the\micount
```

Valores iniciales:
`midim=5.5pt, miskip=10.5pt, micount=3`

Nuevos valores:
`midim=1.83333pt, miskip=2.625pt, micount=0`



Un ejemplo del uso de los comandos anteriores es la definición del siguiente comando, que proporciona la hora en formato hh:mm. Para ello se recurre al comando *\time* que recupera la variable del sistema que almacena el número de minutos que han transcurrido desde la medianoche.

```
\newcount\horas \newcount\minutos
\def\hora{\horas=\time \global\divide\horas by 60
          \minutos=\horas \multiply\minutos by 60
          \advance\minutos by -\time
          \global\multiply\minutos by -1
          \the\horas:\ifnum\minutos<10 0\fi\the\minutos}
```

La última línea tiene por objeto escribir los minutos con dos dígitos (véase la sección 8.6). De este modo, podemos decir que la última compilación de este libro ocurrió el día ‘27 de agosto de 2003’ (\today), a las ‘10:13’ (\hora) horas. La siguiente versión imprime la hora en formato am/pm.

```
\newcount\Hora  
\newcount\horas  
\newcount\minutos  
\def\hora{\Hora=\time \horas=\Hora \divide\horas by 60  
         \minutos=\horas \multiply\minutos by -60  
         \advance\minutos by \Hora  
\ifnum\horas=0\horas=12\else  
         \ifnum\horas>12 \advance\horas by -12 \fi\fi  
\the\horas:\ifnum\minutos<10 0\fi  
\the\minutos\ifnum\Hora>720 pm\else am\fi}
```

Con este nuevo comando, la hora impresa habría sido 10:13am.

8.3. Repitiendo un objeto

ALGUNOS de los ejemplos más usuales donde es conveniente disponer de un comando para repetir un objeto son los índices generales y los índices terminológicos. Si miramos en el índice general de este libro, seguramente aparece una línea como la siguiente:

8.3. Repitiendo un objeto 471

La lnea de puntos ha sido creada por el comando de LATEX \dottedtocline que, entre otras tareas, realiza la siguiente:

\leaders\hbox{\\$\mkern4.5mu\hbox{.}\mkern4.5mu\\$}\hfill}

El código anterior nos sirve para ver cómo debemos utilizar el comando `\leaders`. El usuario debe especificar el carácter u objeto que desea repetir (en el caso anterior es un punto centrado en una caja y separado 4,5 mu de sus extremos) y el espacio que debe ser completado con copias de dicho objeto (en el ejemplo anterior desde la palabra «objeto» hasta el número de página 471). En general, la sintaxis del comando `\leaders` es la siguiente:

\leaders *ObjetoARepetir*\hskip *Longitud*

El *ObjetoARepetir* debe ser una caja mientras que el argumento *Longitud* puede ser cualquier longitud (incluyendo las elásticas), como `1fil`, `1fill`, `50pt`, etc. En lugar de utilizar `\hskip 1fil` o `\hskip 1fill` podemos utilizar, como es sabido, `\hfil` o `\hfill`, respectivamente.

EJEMPLO 8.7

```
\Algunos ejemplos son los siguientes:\\
\null\leaders\hrule\hfill\null\\
\null\leaders\hbox{\hspace{4mm}}\hskip0.7\hskip0.7\hskip0.7\\
\null\leaders\hbox{\hspace{0.7mm}}\hskip0.7\hskip0.7\hskip0.7
```

En el caso de utilizar longitudes elásticas es necesario «marcar» los puntos entre los que actúa el comando `\leaders`; aquí se ha utilizado el comando `\null`, equivalente a `\hbox{}`, aunque también podría haberse utilizado `\mbox{}` o `\kern0pt`

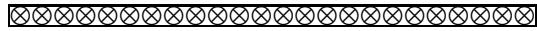
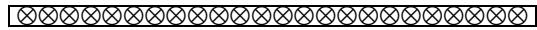
Con el comando `\leaders` puede aparecer un problema cuando la longitud que tenemos que completar con copias no es un múltiplo entero de la caja guía. Cuando la caja que tenemos que repetir es pequeña, esta asincronía puede pasar desapercibida; sin embargo, si el objeto que debemos repetir es grande, el resultado puede no ser bueno. Para solucionar este problema TeX dispone de otros dos comandos para repetir un objeto:

<code>\cleaders ObjetoARepetir\hskip Longitud</code>	<code>\xleaders ObjetoARepetir\hskip Longitud</code>
--	--

El comportamiento de los tres comandos (`\leaders`, `\cleaders`, `\xleaders`) en lo referente al tratamiento del espacio que queda libre después de colocar todas las copias posibles es el siguiente: `\leaders` deja todo el espacio a la derecha, `\cleaders` distribuye dicho espacio entre ambos lados (izquierdo y derecho), mientras que `\xleaders` distribuye el espacio sobrante entre cada copia del objeto. Veamos en el siguiente ejemplo cómo actúan los tres comandos.

EJEMPLO 8.8

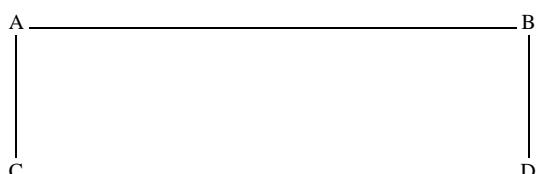
```
\leaders\hbox{$\bigotimes$}\hfill\null
\cleaders\hbox{$\bigotimes$}\hfill\null
\xleaders\hbox{$\bigotimes$}\hfill\null
```



Hasta ahora todos los ejemplos descritos se encargan de copiar la caja guía horizontalmente, de izquierda a derecha. Pero, ¿es posible repetir un objeto verticalmente, de arriba hacia abajo? La respuesta es afirmativa, y la solución consiste en colocar como segundo argumento del comando `\leaders` un desplazamiento vertical (`\vskip`) en lugar de uno horizontal (`\hskip`). La primera aplicación sería definir un nuevo comando `\vrulefill`, dual de `\hrulefill`.

EJEMPLO 8.9

```
\def\vrulefill{\leaders\vrule\vfill\kern0pt}
\box{\offinterlineskip
\hbox to \hsize{A \hrulefill{} B}\vspace{1mm}
\hbox to \hsize{\hbox to 1mm{%
\vbox to 18mm{\vrulefill}\hfil
\vbox to 18mm{\vrulefill}\hfil
\hbox to 1mm{}}\vspace{1mm}
\hbox to \hsize{C \hrulefill{} D}}
```



Repitiendo un objeto verticalmente; observe los signos `%` que hay en el código fuente, que evitan que aparezcan espacios en blanco

8.4. Configurando el diseño de los párrafos

UNA de las tareas para las que \TeX está mejor diseñado es la construcción y composición de párrafos de texto. La mayoría de los procesadores de texto disponen también de funciones para manejar los párrafos, pero ninguno posee la flexibilidad y potencia de \TeX . En esta sección vamos a comentar algunos de los comandos que podemos utilizar para componer los párrafos de una determinada manera.

En el capítulo 1 (pág. 201) ya hemos visto cómo insertar espacios a izquierda y derecha en todas las líneas de un párrafo. De esta forma pueden conseguirse párrafos enteros sangrados por ambos lados (similares a los producidos por los entornos `quote` y `quotation`), «párrafos españoles», etc. En esta sección presentamos algunos comandos adicionales para configurar el diseño de nuestros párrafos.

8.4.1. Modificando el sangrado

Ya hemos visto cómo modificar los espacios que \TeX coloca al principio y al final de cada línea de texto dentro de un párrafo. Sin embargo, muchas veces necesitamos modificar esos espacios sólo para algunas líneas del párrafo. Para llevar a cabo esta tarea \TeX pone a nuestra disposición los siguientes comandos:

<code>\hangindent=Longitud</code>	<code>\hangafter=Núm</code>
-----------------------------------	-----------------------------

donde *Núm* es un número entero: ..., -2, -1, 0, 1, 2... El comando `\hangindent` indica el tamaño del sangrado: si *Longitud* es positiva, el sangrado es en la izquierda, mientras que si es negativa, el sangrado es en la derecha. El comando `\hangafter` indica las líneas a las que afecta dicho sangrado: si *Núm* es negativo, afecta a las primeras *Núm* líneas, mientras que si *Núm* es positivo afecta a las líneas desde el número *Núm*+1 hasta la última.

EJEMPLO 8.10

```
\hangafter=-3\hangindent=2cm
\parfillskip0pt\parindent0pt
La técnica de sangrar sólo las primeras líneas
de un párrafo se utiliza en los libros antiguos
para escribir en ese espacio la primera ... \par
\hangafter=1\hangindent=-2cm
\parindent0pt\parfillskip0pt
Si queremos situar una ilustración (fotografía,
esquema, gráfico, etc.) en la parte inferior
derecha de nuestro párrafo, debemos ... \par
```

La técnica de sangrar sólo las primeras líneas de un párrafo se utiliza en los libros antiguos para escribir en ese espacio la primera letra del párrafo con un tamaño (generalmente grande) y forma especiales. En este libro también se ha utilizado. Si queremos situar una ilustración (fotografía, esquema, gráfico, etc.) en la parte inferior derecha de nuestro párrafo, debemos construir un párrafo como éste, dejando el espacio necesario.

Sangrado de las líneas de un párrafo; observe el truco que hemos utilizado aquí, combinando dos párrafos para dar la apariencia de un solo párrafo, con un doble sangrado

Como se observa, podemos conseguir dejar un espacio en cualquiera de las cuatro esquinas del párrafo: superior izquierda, superior derecha, inferior izquierda o inferior derecha. Para ello debemos asignar valores a `\hangafter` y `\hangindent` según la siguiente tabla:

	<code>\hangindent>0</code>	<code>\hangindent<0</code>
<code>\hangafter>0</code>	inferior izquierda	inferior derecha
<code>\hangafter<0</code>	superior izquierda	superior derecha

8.4.2. Cuando los párrafos son muy especiales

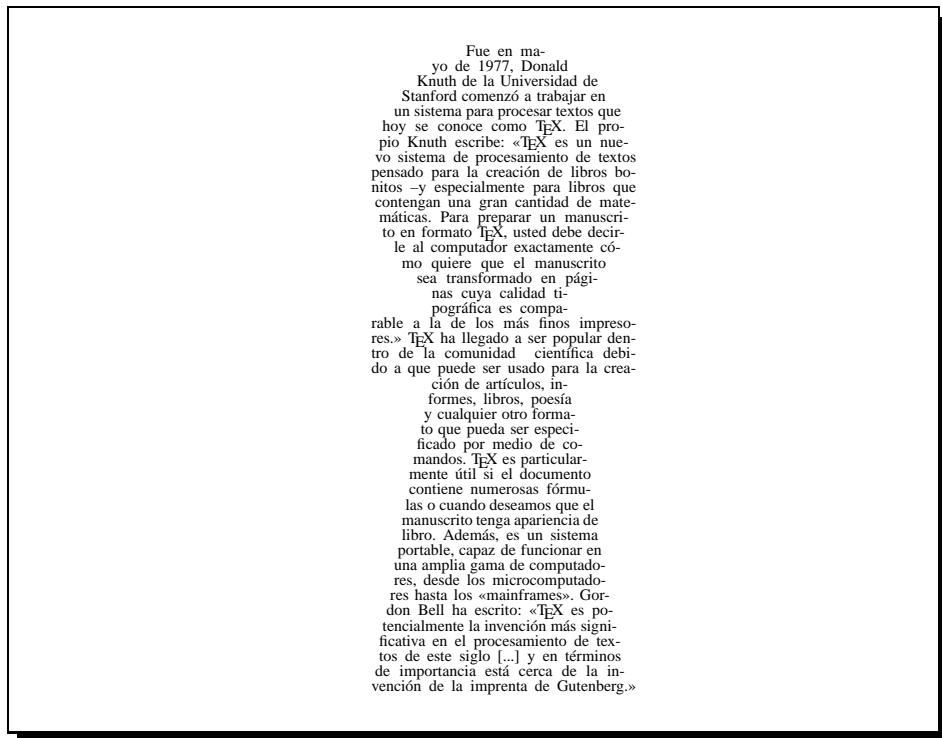


Figura 8.1: Párrafo especial creado con el comando `\parshape`

Todos los comandos anteriores nos proporcionan bastante flexibilidad a la hora de componer párrafos. Pero, ¿qué podemos hacer cuando todo lo anterior no es suficiente? Esto puede ocurrir cuando queremos un párrafo especial, muy especial, como el mostrado en la figura 8.1. Dicho párrafo está construido con el comando `\parshape`, que es el comando más general de \TeX para componer los párrafos. Su sintaxis es la siguiente:

`\parshape=n i1 ℓ1 i2 ℓ2 ... in ℓn`

e indica que las n primeras líneas del párrafo tendrán longitudes ℓ_1, \dots, ℓ_n , respectivamente, y estarán sangradas respecto del margen izquierdo i_1, \dots, i_n , respectivamente. Si el párrafo tiene menos de n líneas, las restantes indicaciones serán ignoradas; por el contrario, si tiene más de n líneas, las condiciones para la n -ésima línea ($i_n \ell_n$) serán repetidas hasta que el párrafo finalice. Para cancelar el efecto de este comando, basta incluir `\parshape=0`.

8.5. Sistematizando tareas

Ya hemos visto cómo dar un formato especial a un párrafo determinado, cómo cambiar el estilo en el que se escribe una fórmula (`\textstyle`, `\displaystyle`), etc. Si los cambios afectan a más de un párrafo o a más de una fórmula, entonces `\TeX` nos proporciona comandos para evitar tener que volver a escribir todos los cambios en cada caso particular.

`\everypar{Comandos}`

Cada vez que `\TeX` ha dado formato a un párrafo, se prepara para componer el siguiente. Pero antes de empezar, ejecuta el comando `\everypar`. Por ejemplo, para saber en cada momento el número de párrafos que llevamos escritos, podemos definir un contador `parrafo`:

```
\newcounter{parrafo}
\setcounter{parrafo}{0}
\everypar{\addtocounter{parrafo}{1}}
```

De este modo, basta con escribir `\theparrafo` para saber en qué párrafo nos encontramos. Concretamente, ahora estamos en el párrafo número 1 desde que hicimos la definición anterior. Para cancelar el efecto, es suficiente con redefinir el comando: `\everypar{}`. Otra aplicación bastante curiosa se ilustra en el siguiente ejemplo, donde formateamos los párrafos de un modo muy especial.

EJEMPLO 8.11

```
\everypar{\parshape 4 0pt \hsize
.07\hsize .93\hsize .1\hsize .9\hsize
.13\hsize .87\hsize}
Otra aplicación muy interesante de la primitiva \everypar
es la que se muestra en este ejemplo. Con una definición
adecuada es posible dar formato a todos los párrafos de
una manera especial.
```

Otra aplicación muy interesante de la primitiva `\everypar` es la que se muestra en ...`\par`. El resultado es que las cuatro primeras líneas del párrafo tienen sangrados diferentes, en orden creciente, y a partir de la cuarta línea se repite. Para acompañar al sangrado, la anchura de las líneas decrece lo mismo.

Realizando una tarea cada vez que iniciamos un párrafo

El siguiente comando tiene el mismo significado que `\everypar` pero se ejecuta cada vez que entramos en modo matemático ordinario, es decir, dentro del texto.

`\everymath{Comandos}`

Supongamos, por ejemplo, que queremos que todas las fórmulas aparezcan en estilo resaltado. Para conseguirlo basta con la declaración `\everymath{\displaystyle}`. De este modo $\sum_{i=1}^n$ produce $\sum_{i=1}^n$ en lugar de $\sum_{i=1}^n$, que sería lo habitual.

`\everydisplay{Comandos}`

Tiene el mismo significado que `\everymath`, pero se ejecuta cada vez que entramos en modo matemático resaltado. Podríamos utilizarlo para escribir, por ejemplo, todas las fórmulas resaltadas en color rojo: `\everydisplay{\color{red}}`.

Finalmente, para sistematizar las actuaciones cuando manipulamos cajas, disponemos de los siguientes comandos:

\everyhbox{Comandos}	\everyvbox{Comandos}
----------------------	----------------------

que se utilizan para ejecutar determinadas acciones siempre que comience una caja horizontal \hbox o vertical \vbox, respectivamente.

8.6. Controlando la situación

EL programa TEX no es un procesador de textos, sino un compilador. Por esta razón, la mayoría de usuarios poco o nada expertos en programación de computadores no son capaces de sacarle todo el provecho. Esto ha influido en el uso masivo de formatos de alto nivel, como LATEX, con los cuales se gana en sencillez pero se pierde en flexibilidad, como ya hemos visto en las secciones anteriores de este capítulo. En la presente sección nos vamos a centrar en un aspecto fundamental en cualquier lenguaje de programación: los condicionales.

Un condicional o estructura de control es una expresión que permite producir diferente material o tomar diferentes acciones dependiendo del valor de una variable lógica. La forma general en un condicional es la siguiente:

IF <Test> [Parte A] ELSE [Parte B] END IF

Si el resultado de la expresión lógica <Test> es verdadero, entonces se procesan las órdenes contenidas en [Parte A], y si es falso, se procesa [Parte B]. Tanto [Parte A] como [Parte B] son opcionales. En TEX, el condicional se escribe como sigue:

\if<Test> [Parte A] \else [Parte B] \fi

Los comandos que nunca deben faltar en un condicional son \if para comenzar y \fi para terminar. Si la [Parte B] no existe, entonces no es necesario incluir \else. Por tanto, son válidos los siguientes condicionales:

\if<Test> \else [Parte B] \fi

\if<Test> [Parte A] \fi

Dentro de [Parte A] o [Parte B] pueden aparecer otros condicionales; es decir, TEX permite la existencia de condicionales anidados propiamente: cada \fi se supone que está asociado con el más reciente \if.

Un consejo: a la hora de escribir condicionales anidados, es conveniente escribirlos de tal forma que visualmente sea fácil distinguirlos. Esto puede hacerse, por ejemplo, sangrando los condicionales anidados:

```
\if<Test1>
  \if<Test2> [Parte A] \else [Parte B] \fi
\else
  \if<Test3> [Parte C] \else [Parte D] \fi
\fi
```

8.6.1. Algunos condicionales de \TeX

En esta sección vamos a comentar algunos de los 17 condicionales que están definidos en \TeX .

`\ifnum Núm1 Relación Núm2`

Se utiliza para comparar números enteros, donde *Relación* puede ser $<$, $>$ o $=$. En el siguiente ejemplo definimos el comando `\tresdig` que siempre escribe un número entero entre 0 y 999 con tres cifras: 001, 057, etc.

EJEMPLO 8.12

```
\def\tresdig#1{\ifnum #1<100 0\fi
  \ifnum #1<10 0\fi #1}
Consideremos la siguiente tabla de valores
obtenida para la función  $f(x)=x^4$ :
\begin{center}\begin{tabular}{cc}
\hline
n & f(n) \\
\hline
1 & 001 \\
2 & 016 \\
3 & 256 \\
\end{tabular}\end{center}
```

Consideremos la siguiente tabla de valores obtenida para la función $f(x) = x^4$:

Imprimiendo los números con al menos tres dígitos

n	$f(n)$
1	001
2	016
3	256

El siguiente condicional sirve para verificar la condición de si un número entero es impar.

`\ifodd Núm`

Recordemos que si queremos analizar el valor almacenado en un contador definido en \LaTeX , debemos sustituir el argumento *Núm* por `\value{NomContador}`, si *NomContador* es el contador con el que queremos trabajar. Por el contrario, si el contador ha sido definido en \TeX (utilizando `\newcount`) entonces debemos utilizar el comando `\the` o `\number` con el contador, es decir, `\the\contador` o `\number\contador`.

EJEMPLO 8.13

Esta página es
`\ifodd\value{page} impar\else par\fi.`

Esta página es impar.

Utilización del condicional `\ifodd`

El siguiente condicional es totalmente análogo a `\ifnum` pero se utiliza para comparar dos longitudes. *Relación* puede ser, como antes, $<$, $>$ o $=$.

`\ifdim Dimen1 Relación Dimen2`

Supongamos, por ejemplo, que deseamos incorporar leyendas al pie de nuestros gráficos, pero queremos que las leyendas se coloquen según el siguiente criterio: si la longitud de la leyenda es menor que la anchura del gráfico, entonces la leyenda se pone centrada en el gráfico; si, por el contrario, la longitud es superior, entonces la leyenda se imprime en estilo párrafo con una anchura igual a la anchura del gráfico. Para realizar esta tarea vamos a construir un comando `\grafico` dependiente de cuatro argumentos con la siguiente sintaxis:

`\grafico{NombreArchivo}{Ancho}{Alto}{Leyenda}`

El funcionamiento del comando es el siguiente:

- Construye una caja de anchura *Ancho* y altura *Alto* para almacenar el gráfico que se encuentra en *Archivo*.
- Calcula la longitud de la *Leyenda*.
- Si la longitud es menor que *Ancho*, entonces la coloca bajo el gráfico y centrada.
- Si, por el contrario, es mayor, construye una caja *\parbox* de anchura *Ancho* para almacenar la descripción.
- Tanto el gráfico como su descripción se incluyen en una caja.

A continuación vamos a escribir el código del comando *\grafico* anterior, donde los ficheros gráficos se incorporan con el comando *\includegraphics* que proporciona el paquete *graphicx*:

```
\newlength{\anchura}
\def\grafico#1#2#3#4{\vbox{\hbox{%
\includegraphics[width=#2,height=#3,keepaspectratio]{#1}}%
\settowidth{\anchura}{#4}\vspace{\abovecaptionskip}%
\ifdim\anchura<#2\hbox to#2{\hss#4\hss}%
\else\hbox{\parbox{#2}{#4}}\fi}}
```

Los siguientes comandos

<i>\ifhmode</i>	<i>\ifvmode</i>	<i>\ifmmode</i>
-----------------	-----------------	-----------------

sirven para verificar si estamos dentro del modo horizontal (normal o restringido), vertical (normal o interno) o matemático (en línea o en estilo resaltado), respectivamente. Por ejemplo, el comando *\ensuremath* proporcionado por *LATEX* está definido esencialmente como sigue:

```
\newcommand{\ensuremath}[1]{\ifmmode #1\else \$#1\$ \fi}
```

El siguiente comando es una generalización del comando *\if*:

<i>\ifcase Núm [AccionesCasoNúm=0]\or [AccionesCasoNúm=1]\or ...</i>
<i>[AccionesCasoNúm=n] \else [AccionesEnOtroCaso] \fi</i>

Su funcionamiento es muy similar al entorno *case* que existe en los lenguajes de programación de alto nivel. Está dividido en varias partes, de las cuales sólo se procesa una en función de un número entero. En total existen $n+2$ partes, de las cuales solamente una se procesa, según el valor de *Núm*: para *Núm=i*, se procesa *AccionesCasoNúm=i*, desde $i=0,1,\dots,n$. Para *Núm* fuera del rango $0-n$ se procesa *AccionesEnOtroCaso*. Un ejemplo de su utilización es la siguiente definición del comando *\today* que imprime la fecha actual (en lugar del comando *\number* se puede utilizar *\the*):

```
\renewcommand{\today}{\number\day{} \ifcase\month\or
Enero\or Febrero\or Marzo\or Abril\or Mayo\or Junio\or
Julio\or Agosto\or Septiembre\or Octubre\or Noviembre\or
Diciembre\fi{} \number\year{}}
```

Otro ejemplo es el siguiente comando *\hexnumber*, que permite expresar un número entero entre 0 y 15 en su versión hexadecimal. Al final de esta sección incluiremos una versión mejorada de este comando, que expresa cualquier número entero en notación hexadecimal (véase la página 487).

```
\def\hexnumber#1{\ifcase\number#1 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or
8\or 9\or A\or B\or C\or D\or E\or F\fi}
```

Construyamos ahora una nueva representación de los contadores equivalente a `\alpha`, sólo que en lugar de utilizar las letras minúsculas del alfabeto latino, utilizará las letras minúsculas del alfabeto griego. Por ello un nombre apropiado puede ser `\greek`:

```
\def\greek#1{\expandafter\@greek\csname c@#1\endcsname}
\def\@greek#1{\ensuremath{\ifcase#1\or \alpha\or \beta\or
\gamma\or \delta\or \varepsilon\or \zeta\or \eta\or
\theta\or \vartheta\or \iota\or \kappa\or \lambda\or
\mu\or \nu\or \xi\or \ο\or \varpi\or \varrho\or \sigma\or
\varsigma\or \tau\or \upsilon\or \varphi\or
\chi\or \psi\or \omega\else\c@ctrerr\fi}}
```

Si observamos la definición anterior nos damos cuenta de la presencia de unos nuevos comandos: `\expandafter`, `\csname` y `\endcsname`, que también son muy interesantes y merecen un comentario, aunque breve. La secuencia «`\csname NombreComando\endcsname`» es equivalente a `\NombreComando`; su utilización está especialmente indicada cuando el comando que queremos ejecutar es una variable y se utiliza conjuntamente con `\expandafter`, que significa: «ejecutar en primer lugar lo que viene detrás del siguiente comando». La sintaxis usual es la siguiente:

```
\expandafter\ComandoA\csname NombreComando\endcsname
```

donde `NombreComando` es el nombre del comando que queremos expandir, el cual suele contener una o más variables. El comando `\expandafter` se suele utilizar también cuando un comando forma parte del argumento de otro comando y es necesario que se expanda previamente (véase el ejemplo 8.17 para una ilustración de este hecho).

El siguiente condicional compara dos argumentos entre sí, dando *verdadero* si son iguales y *falso* en caso contrario.

```
\ifx Arg1Arg2
```

Algunos ejemplos y observaciones son los siguientes:

- Si definimos `\def\a{uno}` y `\def\b{dos}`, entonces el test
Los comandos `a` y `b` son `\ifx\aa\b` iguales `\else` distintos `\fi`
producirá «Los comandos `a` y `b` son distintos».
- Supongamos que tenemos dos comandos `\x` e `\y` que dependen de un argumento y el que se procese uno u otro depende de que `\a` y `\b` sean iguales o distintos. Por ejemplo, supongamos
`\def\x#1{(#1)} \def\y#1{[#1]}`
`\def\z#1{\ifx\aa\b \x{#1} \else \y{#1} \fi}`
Si `\a=\b`, entonces `\z` actúa como `\x`; en caso contrario, actúa como `\y`.
- Los comandos que van a compararse pueden depender de argumentos, pero para que sean considerados iguales por `\ifx` deben depender del mismo número de argumentos. Por ejemplo, consideraremos las líneas siguientes:

```
\def\aa{test}
\def\bb#1{test}
\def\cc#1{test}
\ifx\aa\b iguales \else distintos \fi
\ifx\bb\cc iguales \else distintos \fi
```

El primer test da como resultado «distintos», mientras que el segundo produce «iguales».

- Supongamos que hacemos `\def\aa{5}`. Entonces podemos pensar que `\ifx\aa5` y `\ifx5\aa` van a ser verdaderos; sin embargo, el resultado es falso en ambos casos. Para que la comparación sea correcta debemos guardar el número 5 en un comando (con el mismo número de argumentos que `\aa`). Por ejemplo, supongamos que queremos definir un comando que dependa de un argumento y queremos comprobar al principio si el argumento es vacío o no. Una primera aproximación sería `\ifx#1\empty`, pero el resultado sería siempre falso. El modo correcto de hacerlo es guardar el argumento en un comando y después comparar:

```
\def\comando#1{\def\test{\#1}\ifx\test\empty
               Argumento vacío\else Argumento no vacío\fi}
```

En general, las reglas de comparación que usa el comando `\ifx` para que el resultado de la comparación sea verdadero son las siguientes:

1. Si ambos argumentos son comandos, deben tener el mismo número de argumentos, la misma definición al nivel más alto, es decir, antes de ser expandidos, y el mismo comportamiento respecto de `\long`¹ y `\outer`². Por tanto, dos comandos son considerados iguales si «parecen» iguales en un primer nivel (digamos, al primer vistazo).
2. En otro caso, es decir, si no son comandos, los argumentos comparados deben tener el mismo código de categoría y el mismo código de carácter. A este respecto, un comando tiene código de categoría 16 y código de carácter 256 (véase [32]). Por esta razón la comparación entre un comando y un carácter es siempre falsa.

Algunos ejemplos de aplicación de estas reglas son los siguientes:

- `\ifx AA` es verdadero (regla 2).
- `\ifxA{A}` es falso (regla 2), ya que se compara «A» con «{».
- `\ifxA#1` sirve para comparar si el argumento #1 coincide con «A» o no.

El comando `\ifx` puede utilizarse para comprobar si un determinado comando existe o no (en el sentido de que si no hace absolutamente nada es como si no existiera). Para ello comparamos el comando en cuestión con el comando `\relax`:

```
\ifx\Comando\relax
```

de modo que si el test es verdadero, entonces `\Comando` no ha sido definido.

¹La partícula `\long` precede a la definición, utilizando el comando `\def`, de aquellos comandos que pueden admitir como argumento más de un párrafo, que es lo que sucede siempre con los comandos definidos utilizando `\newcommand`.

²La partícula `\outer` precede a la definición de aquellos comandos que no pueden formar parte del argumento de otro comando y que, por tanto, se pueden considerar «exteriores». Por ejemplo, el comando `\par` está definido con dicha partícula.

8.6.2. Algunos condicionales de L^AT_EX

En L^AT_EX existen numerosos condicionales definidos, que pueden utilizarse junto con los condicionales presentados en la sección anterior. Algunos de estos condicionales se comentan a continuación.

```
\if@twoside \if@twocolumn
```

Son verdaderos si L^AT_EX está procesando el documento con las opciones `twoside` o `twocolumn` declaradas, respectivamente, y falsos en caso contrario. Un ejemplo de la utilización de los condicionales anteriores es la definición del comando `\cleardoublepage`:

```
\def\cleardoublepage{\clearpage
\if@twoside
\ifodd\c@page\else\hbox{}\newpage
\if@twocolumn\hbox{}\newpage
\fi\fi\fi}
```

El siguiente condicional se utiliza en los entornos `eqnarray`:

```
\if@eqnsw
```

Si es verdadero se numera la ecuación, mientras que si es falso la ecuación no va numerada. Así, dentro de la definición del entorno `eqnarray` existe una línea como la siguiente:

```
\if@eqnsw@\eqnnum\stepcounter{equation}\fi
```

que determina si la ecuación va a ir numerada o no. En caso afirmativo, imprime la etiqueta que contiene el número de la ecuación (`\@eqnnum`) e incrementa el contador de las ecuaciones (`\stepcounter{equation}`).

```
\@ifnextchar Carácter{ParteA}{ParteB}
```

Este comando está construido sobre el comando `\ifx`. Se utiliza cuando queremos analizar el siguiente carácter para, en función de dicho análisis, procesar unas órdenes u otras. Este condicional es muy utilizado en L^AT_EX en los comandos que admiten argumentos opcionales, caracterizados por ir entre corchetes. Por ejemplo, supongamos que queremos definir un comando que dependa de un argumento y cuya misión sea recuadrar dicho argumento con una línea de grosor 0,4 pt. Opcionalmente podemos especificarle un primer argumento (entre corchetes) para indicar otro grosor alternativo. Una posible definición, utilizando el comando `\fbox` de L^AT_EX, sería la siguiente:

```
\def\mirecuadro[#1]{\fboxrule#1\fbox{#2}}
\def\recuadro{\@ifnextchar[\mirecuadro]{\mirecuadro[0.4pt]}}
```

Entonces `\recuadro{test A}` producirá `[test A]`. Si queremos que el recuadro tenga 1 pt de grosor, debemos escribir `\recuadro[1pt]{test B}` que producirá `[test B]`.

El siguiente comando, como el anterior, está construido sobre el comando `\ifx`:

```
\@ifundefined{NombreComando}{ParteA}{ParteB}
```

Es equivalente a `\ifx\NombreComando\relax ParteA\else ParteB\fi`. Por tanto, si el comando `\NombreComando` no existe se ejecuta la `ParteA`, y en caso contrario la `ParteB`.

8.6.3. Definición de nuevos condicionales

En \TeX se pueden definir nuevos condicionales con el comando \newif . La sintaxis es:

```
\newif\iftest
```

donde \iftest es el nombre del nuevo condicional. Tras una línea como la anterior, \TeX define tres nuevos comandos: \iftest , \testtrue y \testfalse . A partir de este momento, es posible escribir cosas como las siguientes: \testtrue (para asignar a la variable test el valor de verdadero), \testfalse (para asignar a la variable test el valor de falso), $\text{\iftest...else...fi}$ (para realizar un condicional).

Supongamos que queremos modificar el comando \grafico definido en la página 478 para que permita, opcionalmente, poner un recuadro al dibujo. Para ello vamos a definir un nuevo condicional \ifrecuadrar en función del cual procederemos a recuadrar o no el dibujo. Una posible solución sería la siguiente:

```
\newlength{\anchura}
\newif\ifrecuadrar
\recuadrafalse %se inicializa a FALSO
\def\grafico[#1#2#3#4#5]{%
\ifx#5R\recuadratrue\else\ifx#5r\recuadratrue\fi\fi
\vbox{\ifrecuadrar\else\fboxrule0pt\fi\fboxsep0pt
\hbox{\fbox{\includegraphics[width=#2,height=#3,%
keepaspectratio]{#1}}}}%
\vspace{\abovecaptionskip}\settowidth{\anchura}{#4}%
\ifdim\anchura<#2 \hbox to#2{\hss#4\hss}\else
\hbox{\parbox{#2}{#4}}\fi}}
```

El valor predefinido de la variable \ifrecuadrar es `false`, por lo que los gráficos no se recuadrarán. Lo primero que hacemos en el comando \grafico es comprobar si el último argumento es `R` o `r`, para cambiar el valor de la variable \ifrecuadrar a `true`. El comando siempre utiliza \fbox , sólo que si no se debe recuadrar, entonces el recuadro lo pinta con un grosor de `0pt`. El comando \grafico descrito anteriormente puede ser mejorado si el argumento que se utiliza para recuadrar se hace opcional, de manera que sólo tengamos que indicarlo si queremos que nuestro gráfico esté recuadrado. Una posible versión es:

```
\def\migrafico[#1]#2#3#4#5{%
\ifx#1R\recuadratrue\else\ifx#1r\recuadratrue\fi\fi
\vbox{\ifrecuadrar\else\fboxrule0pt\fi\fboxsep0pt
\hbox{\fbox{\includegraphics[width=#3,height=#4,%
keepaspectratio]{#1}}}}%
\vspace{\abovecaptionskip}\settowidth{\anchura}{#5}%
\ifdim\anchura<#3 \hbox to#3{\hss#5\hss}\else
\hbox{\parbox{#3}{#5}}\fi}}
\def\grafico{\@ifnextchar[\{\migrafico\}{\migrafico[]}}
```

En la última línea del código anterior hemos escrito `\migrafico[]`. En su lugar podríamos haber escrito `\migrafico[x]`, donde *x* puede ser cualquier carácter distinto de r y R.

Otra posibilidad es definir el comando `\grafico` dependiendo de si el siguiente carácter es un asterisco o no, lo que implicará que el gráfico se recuadre o no, respectivamente. La forma de hacerlo podría ser la siguiente:

```
\def\migrafico*#1#2#3#4{%
  \vbox{\hbox{\fboxsep0pt\fbox{%
    \includegraphics[width=#2,height=#3]{#1}}}}%
  \vspace{\abovecaptionskip}\settowidth{\anchura}{#4}%
  \ifdim\anchura<#2 \hbox to#2{\hss#4\hss}\else
    \hbox{\parbox{#2}{#4}}%
  \fi}%
\def\grafico{\@ifnextchar*{\fboxrule0.4pt\migrafico}{%
  \fboxrule0pt\migrafico*}}
```

8.6.4. Los bucles

TeX puede realizar bucles mediante los siguientes comandos:

`\loop ParteA \if... ParteB \repeat`

donde *ParteA* y *ParteB* son cualquier sucesión de comandos, y donde `\if...` es cualquier condicional (sin la correspondiente partícula `\fi`). *TeX* procesa primero la *ParteA*; luego, si la condición es verdadera, procesa la *ParteB* y repite el proceso de nuevo comenzando por la *ParteA*. Si la condición es falsa en alguna etapa de este proceso, el bucle finalizará y *TeX* seguirá procesando los comandos que vengan a continuación.

En el siguiente ejemplo utilizamos un bucle para crear un comando `\numeros`, dependiente de un argumento (que debe ser un número entero positivo), cuya misión es crear la sucesión de los números enteros positivos menores que dicho argumento.

EJEMPLO 8.14

```
\newcount\minum
\def\numeros#1{\ifnum#1<2 1%
  \else 1\minum=1\loop \advance\minum by1%
    \ifnum\minum<#1, \the\minum\repeat\fi}
Los números ... menores que 50 son: \numeros{50}.
```

Los números enteros positivos menores que 50 son: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49.

`| Definición de un bucle mediante los comandos \loop...\repeat`

Para condicionales específicos relacionados con los números y las longitudes, *LATeX* tiene definidos los siguientes bucles:

`|@whilenum{TestNum}\do {Acción}`

`|@whiledim{TestDim}\do {Acción}`

En el primero de ellos se evalúa una relación numérica, y mientras la relación *TestNum* sea verdadera se procesará el argumento *Acción*. En el segundo comando se evalúa una relación entre

dimensiones, procesando la *Acción* siempre que la relación *TestDim* sea verdadera. En el siguiente ejemplo utilizamos el bucle `\@whilenum` para crear la sucesión de todos los números pares menores que un número dado.

EJEMPLO 8.15

```
\newcount\minum
\makeatletter
\def\pares#1{\minum=2\@whilenum\minum<#1\do
    {\the\minum, \advance\minum by2}}
\makeatother
Los números pares menores que 125 son: \pares{125}
```

Los números pares menores que 125 son: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124,



Para bucles en los que la condición de control no es una relación entre números o longitudes, sino el valor de un condicional de *T_EX*, *L_AT_EX* dispone del siguiente comando:

`\@whilesw\iftest\fi{Acción}`

donde `\iftest` es un condicional definido. *T_EX* procesará ininterrumpidamente los comandos de la *Acción* hasta que el condicional `\iftest` sea falso.

Otro tipo de bucles habituales en los lenguajes de programación son las estructuras «for/next», en las cuales se ejecuta una serie de acciones para cada uno de los elementos de una lista, que habitualmente son números, aunque también pueden ser de otro tipo. *L_AT_EX* nos proporciona el siguiente comando para trabajar con este tipo de estructuras:

`\@for\nombre:=\lista\do{Acción}`

donde `\lista` es un comando cuyo valor es una lista de elementos separados entre sí por comas y `\nombre` es una variable (que no tiene por qué estar previamente definida) que irá almacenando los diferentes elementos de la lista. Para cada uno de estos valores, *T_EX* procesará los comandos de la *Acción*, y cuando la lista haya sido agotada, entonces *T_EX* seguirá procesando los comandos que vengan a continuación.

En el siguiente ejemplo calculamos la longitud (número de caracteres no blancos) de cada una de las palabras que aparecen en una lista y construimos una tabla con las palabras y sus longitudes. El comando `\longitud` que utilizamos está descrito en el ejemplo 8.17.

EJEMPLO 8.16

```
\def\lista{Antonio,Bernardo,José Manuel,Pascual,Salvador}
\begin{flushright}\begin{tabular}{l}
Nombre y longitud\\\hline
\makeatletter
\@for\nombre:=\lista
\do{\hbox to 20mm{\nombre\hss}\longitud{\nombre} \\ }
\makeatother
\end{tabular}\end{flushright}
```

Nombre y longitud	
Antonio	7
Bernardo	8
José Manuel	10
Pascual	7
Salvador	8



Podemos utilizar las mismas ideas que en el comando `\numeros` (definido en el ejemplo 8.14) junto con el comando `\@for` para definir bucles «for/next» ordinarios, es decir, bucles que se

repiten para todos los valores numéricos de una variable entre un valor mínimo y un valor máximo, con un cierto incremento de la variable en cada etapa del proceso. En primer lugar, definimos un comando para crear la lista de números:

```
\CrearLista{NombreLista}{Inicial}{Final}{Salto}
```

Después de ejecutar `\CrearLista`, el comando `\NombreLista` será la lista de los números naturales comprendidos entre el valor *Inicial* y el valor *Final*, con incrementos iguales a *Salto*. El código de dicho comando es el siguiente:

```
\def\CrearLista#1#2#3#4{\newcount\nn \nn=#2
  \def\Lista{#2}
  \loop \advance\nn by #4
    \ifnum\nn<#3 \edef\Lista{\Lista,\number\nn}
  \repeat
  \expandafter\let\csname #1\endcsname\Lista}
```

Ahora, un clásico bucle for/next puede ser construido, por ejemplo, como sigue:

```
\CrearLista{milista}{1}{20}{2}
@for\num:=\milista\do{...}
```

donde «...» sirve para indicar la acción a realizar en cada etapa del bucle.

8.6.5. Más ejemplos

Contar las letras de una palabra

En este ejemplo definimos un comando que calcula el número de caracteres no blancos que aparecen en una cadena de texto.

EJEMPLO 8.17

```
\newcount\nn
\def\longitud#1{\nn=0\expandafter\contar#1\end
  \number\nn}
\def\contar#1{\ifx#1\end\let\next=\relax
  \else\advance\nn by1 \let\next=\contar\fi\next}
La longitud de la palabra «esternocleidomastoideo»
es \longitud{esternocleidomastoideo}...
```

La longitud de la palabra «esternocleidomastoideo» es 22.

La frase “La longitud de la palabra «esternocleidomastoideo» es” contiene 49 caracteres no blancos.

El comando `\end` se utiliza únicamente como delimitador para determinar cuándo finaliza el argumento

Ocultar texto

Vamos a construir un entorno para ocultar texto, siempre que una variable lógica almacene el valor verdadero. La idea de la definición consiste en abrir una caja y almacenar en ella todo lo que el usuario escriba dentro del entorno. Una vez finalizado el entorno se procede a comprobar el valor de la variable lógica y, en caso de que éste sea falso, se «deshace» la caja y se imprime el contenido. El código es el siguiente:

```
\newbox\boxtoculto
\newif\iftoculto
\tocultottrue
\newenvironment{textoculto}
  {\setbox\boxtoculto\vbox\bgroup}
  {\egroup\iftoculto\else\par\unvbox\boxtoculto\fi}
```

Una posible utilización aparece en el ejemplo 8.18, donde se puede observar que se pueden anidar distintos entornos `textoculto`. El comando `\setbox` sirve para almacenar el contenido del entorno en la caja `\boxtoculto`, a la que se le da formato en modo vertical. Para no tener problemas con los saltos de página, antes de imprimir el contenido (si éste es el caso) «deshacemos» la caja mediante el comando `\unvbox`.

Una posible aplicación interesante de este entorno es la siguiente. Imaginemos un documento del que vamos a realizar dos versiones: una completa y otra reducida, en la que algunas partes de la versión completa no aparecerán (por ejemplo, un examen con y sin soluciones). No es una buena idea (de hecho, es pésima) escribir dos ficheros independientes, ya que esto obliga a un doble mantenimiento. Lo conveniente es situar las partes que deben suprimirse en la versión reducida dentro de un entorno `textoculto`, de tal forma que con sólo indicar `\tocultottrue` o `\tocultofalse` en el preámbulo del documento podamos hacer desaparecer o aparecer todos los textos que están en un entorno `textoculto`, consiguiendo así un cambio global con un mínimo esfuerzo.

Estas ideas pueden generalizarse, de modo que en un documento podemos tener varios tipos de texto oculto, por ejemplo, `textocultoA`, `textocultoB`, `textocultoC`... De esta manera podemos generar múltiples documentos distintos, todos ellos a partir de un único documento completo, con las ventajas que esto supone.

EJEMPLO 8.18

```
\textocultofalse
\begin{textoculto}
La luna está relativamente cerca, a 384\,392
kilómetros de la Tierra. ...
\textocultottrue
\begin{textoculto}
Además, ambos cuerpos tienen la misma edad:
entre 4\,500 y 4\,600 millones de años.
\end{textoculto}
\end{textoculto}
```

La luna está relativamente cerca, a 384 392 kilómetros de la Tierra. Su diámetro es de 3 476 kilómetros, poco más de la cuarta parte de su planeta de referencia.

Un entorno para ocultar texto

Invertir una palabra

Veamos a continuación un ejemplo curioso. Vamos a construir un comando que toma una palabra (sin comandos ni espacios en blanco) como argumento y devuelve la misma palabra pero con las letras en orden inverso. El comando utiliza ideas similares a las del ejemplo 8.17; la clave está en que cada carácter no blanco leído se va almacenando en orden inverso a como se va leyendo.

EJEMPLO 8.19

```
\def\Invertir#1{\def\INV{} \INVCAD#1\end\INV}%
\def\INVCAD#1{\ifx#1\end\let\next=\relax
  \else\CONCAD#1\let\next=\INVCAD\fi\next}%
\def\CONCAD#1{\edef\INV{#1\INV}}
\begin{center}
  \Invertir{Espejo}Espejo\par
  Excitante\Invertir{Excitante}\par
  \Invertir{Sorprendente}Sorprendente
\end{center}
```

El comando `\end` se utiliza únicamente como delimitador para determinar cuándo finaliza el argumento

Notación hexadecimal

El siguiente ejemplo es una versión mejorada del comando `\hexnumber` que aparece en la página 478, ya que transforma cualquier número entero en notación hexadecimal (basado en el comando `\hex` que aparece en la página 219 de [32]).

EJEMPLO 8.20

```
\newcount\nn \newcount\pp \newcount\qq
\def\hexnumber#1{{\pp=1 #1 \divide\nn by 16
  \ifnum\nn>0\hexnumber{\the\nn}\fi
  \qq=\nn \multiply\qq by -16
  \advance\pp by \qq \hexdígito}
\def\hexdígito{\ifnum\pp<10 \number\pp\else
  \advance\pp by -10 \advance\pp by 'A \char\pp \fi}
\begin{center}
  \textbf{Tabla de conversión de algunos números:}\vspace{[5mm]}
\begin{tabular}{r r}
Dec. & Hex. \\
\hline
12 & C \\
15 & F \\
30 & 1E \\
123 & 7B \\
456 & 1C8
\end{tabular}
\end{center}
```

Tabla de conversión de algunos números:

Observe que la definición del comando es *recursiva*, de modo que múltiples copias del mismo pueden estar activas simultáneamente. Por esta razón son necesarias las dobles llaves de apertura y cierre, para evitar que unas instancias del comando interfieran en las otras

PARA SABER MÁS

- Hemos visto cómo hacer aritmética en \TeX utilizando los comandos primitivos `\advance`, `\multiply` y `\divide`. De éstos, únicamente `\divide` no tiene ningún comando equivalente en \LaTeX . A pesar de ello, para realizar operaciones sencillas con estos comandos debemos escribir muchas líneas de código. Para facilitar los cálculos en \LaTeX , K.K. Throop y F. Jensen han desarrollado el paquete `calc` [66], que introduce la aritmética real, más habitual y fácil de entender (véase [11] para una descripción de este paquete).

- ▶ El paquete calc se limita a las cuatro operaciones básicas. M. Mehlich ha desarrollado el paquete fp [45], que permite realizar aritmética en coma flotante, no limitándose a las operaciones básicas y proporcionando comandos para el cómputo de las funciones reales más usuales (véase [11] para una descripción de este paquete).
- ▶ Pero, ¿aún necesita saber más? Seguro que su sueño es llegar a ser un TEXwizard. Sólo podemos remitirle al origen de todo esto: «The TEXbook» [32], del creador Donald Ervin Knuth.

Cómo hacerlo

En la página 234 se ha explicado cómo, al elaborar un índice terminológico, se puede facilitar la lectura multidireccional de las entradas mediante el uso de mayúsculas en las iniciales de algunas subentradas. Si usted no conoce esta regla consulte la página mencionada.

A

- abstract, 54
- acento(s), 35
 - en fórmulas, *v. fórmulas*
- ACROBAT, Acceso a los menús de, 411
- actas, 49
- agradecimientos, 53
- agrupar celdas en tablas, *v. tablas*
- alinear
 - a derecha o a izquierda, *v. texto alineado*
 - cajas, *v. cajas*
 - ecuaciones, 385
 - en columnas, 104
- altura
 - de cajas, *v. cajas*
 - de páginas, *v. página(s)*
- anchura
 - de cajas, *v. cajas*
 - de la línea en curso, *v. página(s)*
 - de páginas, *v. página(s)*
- anexo, 56
- animaciones en pantalla, 459
- antetítulo, *v.t. título*
 - de las partes, 51, 205
 - de los apéndices, 56
 - de los capítulos, 51, 57, 205
- añadir elementos al fondo, 448
- apaisado, *v. página(s)*
- apartados, Listas con, *v. listas*
- apéndices, 56
 - Antetítulo de los, *v. antetítulo*
 - Numeración de los, 56
- archivo de transiciones, 446

argumento(s), 13

- Comparación de, 479
- Delimitadores de los, 464
- en definición de comandos, 94, 95, 464
- en definición de entornos, 95
- móviles, 92, 208
 - Comandos frágiles en, *v. comandos frágiles*
- obligatorio, 91
- optativo, 91

arracada, 310

- artículos, 49
- asignación de niveles a una subpágina, 450
- atajos de teclado, *v. taquigrafía*
- autor(es), *v.t. documento*
 - de PDF, 409
 - de la presentación, 436, 456
 - del documento, 53

B

- bandera, *v. texto*
- barra de navegación, 440
 - Activar y desactivar la, 440
 - Colores de la, 441
 - en HTML, 430
 - Personalizar la, 441
- bibliografía, *v. citas bibliográficas*
- blanco, *v. espacio*
- borde, *v. cajas*
- borrador
 - de la presentación, 455
 - del documento, 60
 - Señal de línea larga en el, 230
- BoundingBox, *v. gráficos, Dimensiones de los bucles*, 483
 - controlados por dimensiones, 483

bucles (cont.)

- controlados por números, 483
- controlados por una lista, 484
- controlados por variables, 349
- del tipo for/next, 484

C

cabecera de las páginas, *v. marcas*
cajas

- \TeX sólo maneja, 315
- Almacenar, 152, 323
 - muy grandes, 153
- como pequeñas páginas, 148
- con color, *v. color(es)*
- de altura prefijada, 148, 320
- de anchura prefijada, 147, 148, 318
- Definir, 152
 - sencillas, 147
- Desplazar
 - horizontalmente, 321
 - verticalmente, 151, 321
- Dimensiones de las, 323
- horizontales, 317
- marginadas, 310
- Posicionar
 - a la derecha, 318
 - a la izquierda, 318
 - en el centro, 318
- Recuadrar, 147
 - con doble marco, 148
 - con sombra, 148
 - con un óvalo, 148
 - Grosor del marco al, 147
 - Separación del marco al, 147
- Reutilizar, 152, 323
- Separar, 146, *v.t. longitud(es)*
- Sobreimprimir, 327
 - verticales, 319
- calderón (¶), 38, 39
- capítulos, 50
 - Antetítulo de los, *v. antetítulo*
 - Dónde se inician los, 61
- carácter, *v. fuente(s)*
- caracteres reservados, 31
 - en índices, *v. índice(s) terminológico(s)*
- carta, 49, 154
- centrar, *v. texto centrado*
- circunferencia que rodea a un carácter, 38
- cita textual, 27
- citas bibliográficas, 123
 - Agrupar, 266
 - Alfabetización de, 251

citas bibliográficas (cont.)

- Bases de datos para BIB \TeX de, 256–266
- Mayúsculas y minúsculas en las, 261
- Preámbulos en las, 264
- Abreviaturas en las, 262
- Autor y editor en las, 260
- Campos en las, 259
- Imprimir todos los registros de las, 252
- Mantenimiento de las, 264
- Referencias en las, 263
- Registros en las, 261
- como hiperenlaces, 125, *v.t. hipertexto*
- Estilos para BIB \TeX de
 - Adaptar, 265
 - adicionales, 257
 - estándar, 253
- Generar la lista de
 - con BIB \TeX , 251
 - con BIB $\text{\TeX}8$, 251
 - manualmente, 123
 - opción openbib, 255
- Títulos de la lista de, *v. título*
- Varias listas de, 268
- clases de documento, *v. documento*
- codificación, *v. páginas de códigos*
- código fuente, *v. texto fuente*
- color(es)
 - Cajas con
 - de fondo, 275
 - en el marco, 275
 - Controladores para, 45, 274
 - de fondo
 - en la página, 275
 - para una caja, 275
 - Restaurar a blanco el, 275
 - de la página, 448
 - Degrado horizontal del, 448
 - Degrado vertical del, 448
 - Definir, 274
 - Desactivar el, 273
 - disponibles, 46
 - en hipertexto, 75, 406
 - en tablas, *v. tablas*
 - Especificar, 275
 - Modelos de, 274
 - Nombres de, 46
 - Texto en, 46
- columnas
 - Alinear en, 104
 - Anchura de las, *v.t. página(s)*

- columnas (cont.)
 - en tablas, 112
 - Línea de separación entre, 102
 - periodísticas, 102
 - Separación entre, 102
 - Una o dos, 61
- coma decimal, 139, 392
- comando(s), 13, 91
 - Ampliar el campo de acción de un, 466
 - Argumentos de un, 91, 464
 - Campo de acción de un, 465
 - con @, 217
 - constante, 467
 - Definir un, 94, 95, 463, 464
 - constante, 467
 - global, 466
 - mediante expansión, 467
 - recursivamente, 467
 - Expansión de los argumentos de un, 467
 - Expansión de un, 93
 - frágiles, 208
 - Problemas con, 208
 - Proteger, 208
 - Nombre de un, 90
 - Redefinir un, 94, 95
 - robustos, 208
 - Sacar una copia de un, 468
- comentarios, *v. texto fuente*
- comillas
 - de abrir y cerrar (« ‘ ’ »), 35
 - de seguir, 175
 - Uso de las, 36
- comparación
 - de argumentos, 479
 - de longitudes, 477
 - de números enteros, 477
- compilación, 4
 - a trozos, 209, 210
 - guardando información, 210
- Errores de, 19
 - Causas posibles de los, 19–21
- interactiva, 19
- Modos de, 19
 - sin interrupciones, 19
- Visionar la, 5, *v.t.* visores
- componer, *v. compilación*
- condicionales, 476
 - Definir nuevos, 482
 - para comparar argumentos, 479
 - para comparar longitudes, 477
 - para comparar números enteros, 477
- condicionales (cont.)
 - para conocer el siguiente carácter, 481
 - para detectar el modo de trabajo, 478
 - para saber si un comando existe, 481
 - para saber si un número es impar, 477
- conectar objetos con una curva, 350
- contador(es)
 - Aumentar el valor de un, 469
 - Conocer el valor de un, 141, 195
 - Crear nuevos, 195, 468
 - de capítulos, 139, 204
 - de pies de página, 105
 - en minipage, 150
 - de páginas, 139
 - de secciones, 139, 204
 - Disminuir el valor de un, 469
 - Dividir un, 470
 - Formato de un, 139
 - Incrementar para referencias cruzadas, 196
 - Manipulación local de, 466
 - Modificar el valor de un, 141
 - Multiplicar un, 470
 - Representación de un, 140
 - Subordinar, 195, 388
- contar caracteres no blancos, 485
- «continúa en la página siguiente...», 223
- controladores
 - para el color, *v. color(es)*
 - para gráficos, *v. gráfico(s)*
 - para hiperenlaces, *v. hipertexto*
- convertir
 - de LATEX a HTML, 417
 - ficheros gráficos, 333
 - texto en gráfico, 427
- copyright (©), 38
- corrección itálica, 43
- crear archivo(s)
 - HTML con TEX4HT, 417
 - PDF
 - con PDFLATEX, 8
 - desde DVI, 8
 - desde PS, 8
 - PS con DVIPS, 7
- cuadros, 118
 - Leyenda de, 118
 - Rotar, 340
 - Ubicación de, 118
 - Índice de, 118
- cuerpo del documento, *v. texto fuente*

D

dedicatoria, 54
 definir
 un comando, 94, 95, 463, 464
 constante, 467
 un contador, 468
 un entorno, 95
 una longitud
 elástica, 468
 rígida, 468
 delimitadores de argumentos, 464
 demostración, *v.* teorema(s)
 determinantes, *v.* fórmulas
 diagramas, 352
 comutativos, *v.* fórmulas
 dibujar
 con PSTricks, *v.* PSTricks
 cuadrículas, 346
 círculos, elipses y arcos, 345
 líneas y poligonales, 343
 líneas, *v.* filete(s)
 polígonos y rectángulos, 345
 puntos y curvas, 343
 rayas, *v.* filete(s)
 dividir
 palabras, *v.* palabra(s)
 un contador, 470
 un documento, *v.* compilación
 una longitud, 470
 doble espacio, *v.* interlínea
 documento
 a dos columnas, 61
 Autores del, 53
 Clases de, 49
 actas, 49
 artículos, 49
 cartas, 49
 informes, 49
 libros, 49, *v.t.* libro(s)
 Opciones de las, *v.* opciones
 transparencias, 49
 Estructura de un, *v.* unidades de estructura
 Fecha del, 53
 fuente, *v.* texto fuente
 Título del, 53
 Notas a pie en el, 53, 106
 Página del, 54, 61
 Versiones preliminares del, 60
 dólar (\$), 32
 «driver», *v.* controladores
 DVI, *v.* salida, Formato de

E

ecuaciones, *v.* fórmulas
 ele geminada, 173
 encuadernación, *v.* margen
 enfatizada, *v.* letra(s)
 enlace(s)
 con la primera subpágina, 449
 de hipertexto, *v.* hipertexto
 enmarcar
 fórmulas, *v.* fórmulas
 listas, *v.* listas
 material centrado, *v.* alinear
 página, *v.* página(s)
 entorno(s), 13, 92, 95
 Argumentos de, 92
 Configurar
 en HTML, 429
 Definir un, 95
 flotantes
 Leyendas en, *v.* leyendas
 Parámetros de los, 302
 Redefinir un, 95
 epígrafe
 de cuadros, *v.* cuadros, Leyenda de
 de figuras, *v.* figuras, Leyenda de
 errores, *v.* compilación
 escala, Modificar la, 339
 espacio(s)
 de la interlínea, *v.* interlínea
 después de signos de puntuación, 37, 160
 después de un comando, 29
 elástico, *v.* longitud(es)
 en fórmulas, *v.* fórmulas
 entre palabras, 22
 Añadir, 24
 horizontal, 68, 134
 Ignorar los, 202
 Respetar los, 202
 rígido, *v.* longitud(es)
 Signo para indicar un, 38
 vertical, 68
 estilo(s)
 de bibliografía, *v.* citas bibliográficas
 de presentación, 454, 455
 de página, 64, *v.* página(s)
 en presentaciones, 440
 de teorema, *v.* teorema(s)
 de índices, *v.* índice terminológico
 Ficheros de, *v.* paquetes
 estructura, *v.* unidades de estructura
 estructuras de control, *v.* condicionales

etiqueta(s), 72
en listas, *v. listas*
Errores usuales al usar, 73
para referencias, 71
Precauciones con el uso de, 73
euro (€), 39, 186, 192
expansión de un comando, 93
exponentes, *v. fórmulas, superíndices*

F

familia de tipos, *v. fuentes*
fecha
de un documento, *v. documento*
Imprimir la, 15
fichero(s)
DVI, 4
HTML, 417
PDF, 7
Características de, 7
Generar, *v. crear archivo(s)*
Visores de, *v. visores*
PS, 6
Características de, 6
Generar, *v. crear archivo(s)*
Visores de, *v. visores*
utilizados en un documento, 211
figuras, 84
en texto a dos columnas, 86
Leyenda de, 85
marginadas, 310, *v.t. cajas*
Rotar, 340
Ubicación de, 85
Índice de, 86
filas en tablas, 111
filete(s)
caña, 150
Crear, 150
en tablas, *v. tablas*
extensibles, 153
horizontales, 322
Longitud de, 150
media caña, 150
verticales, 322
fligrina, 327, 438
Modificar la, 439
flotantes, *v. entornos flotantes*
folio, *v. marcas*
fondo de la página
Añadir elementos al, 448
Color del, 448
formato, *v. salida*

fórmulas
Acentos en, 129, 368
de varias líneas, 380
Agrupar ecuaciones en, 385
Alinear ecuaciones en, 385
Formar bloques de ecuaciones en, 386
Partir ecuaciones en, 383
Salto de página en, 388
Definición de, 127
Delimitadores en, 376
Ajustar tamaños de, 377
determinantes, 378
diagramas commutativos, 395
en color, 129
en HTML, 429
enmarcadas, 328, 371
Espacios en, 129, 133, 372
flechas, 365
debajo de símbolos, 373
encima de símbolos, 373
fracciones, 131
continuas, 375
genéricas, 375
funciones, 389
integrales, 392
múltiples, 394
llaves, 32, 376
debajo de símbolos, 372
encima de símbolos, 372
matrices, 378–379
numeradas, 128
Modificar etiquetas en, 382
Modificar numeración en, 388
Suprimir numeración en, 381, 382
Ubicación de la etiqueta en, 359
números combinatorios, 132
Operadores de tamaño variable en, 392
Definición de, 394
Límites en, 393
Parámetros globales de las, 400
Puntos suspensivos en, 369
químicas, 135
raíces, 130
Referencias a, 128
Saltos de línea en, 129
Subrayar, 372
subíndices, 130
en varias líneas, 393
sumatorios, 392
superíndices, 130
en varias líneas, 393

- fórmulas (cont.)
- símbolos, *v.* símbolos matemáticos
 - tamaño, 132, 370
 - de las fuentes, 129
 - teoremas, *v.* teorema(s)
 - Textos en, 133
 - Unos símbolos sobre otros en, 372
- fracciones, *v.* fórmulas
- fuente, *v.* texto fuente
- fuente(s), *v.t.* letra
- Atributos de las
 - familia, 41
 - grosor o series, 43
 - perfil, 42
 - tamaño, 44
 - caligráficas mejoradas, 365
 - cm-super, 181
 - Codificación de, 179
 - Computer Modern, 177
 - en formato PostScript, 181
 - Ficheros asociados a, 179
 - de símbolos, 189
 - EC, 179
 - en formato PostScript, 181
 - en modo matemático, 362
 - HTML y XHTML, 423
 - imprimir
 - tablas, 187
 - un carácter, 189
 - PostScript, 181
 - de Adobe, 182, 184
 - URW, 187
 - Página de códigos de, 179
 - Seleccionar una, 193
 - True Type, 192
- funcionamiento de L^AT_EX
- Esquema completo de, 6
 - Esquema simplificado de, 3
- G**
- girar, *v.* rotar
- glosarios, 248, 270
- Contrucción de
 - con BIBT_EX, 271
 - con MAKEINDEX, 249
- grabados, *v.* gráficos
- grados, Signo para los, 38
- gráfico(s)
- Controladores para inclusión de, 80
 - Convertir, 333
 - de EPS a PDF, 334
- gráfico(s) (cont.)
- de PS a EPS, 334
 - de PS a PDF, 334
 - de otros formatos a EPS, 334
 - Crear con PSTRicks, *v.* PSTRicks
 - Dimensiones de los, 81
 - en HTML, 422, 427
 - Estrategias para inclusión de, 332
 - formatos incorporables, 80–82
 - Imprimir sólo una parte del, 336
 - Inclusión de, 82, 335
 - Parámetros para la, 82
 - Rotar, 337
 - Tipos de, 79
- grosor
- de los filete(s), *v.* filete(s)
 - de los tipos, *v.* fuente(s)
- grupo(s)
- Definir un, 89
 - ¿qué son?, 89
- guiones
- de división silábica, *v.* palabras
 - más ayuda, 230
 - que no parten expresiones, 370
 - Tipos de, 36
- H**
- hexadecimal, Notación, 487
- hipertexto
- Acceso a los menús de ACROBAT con, 411
 - Configurar los colores de, 406
 - Configurar los enlaces de, 405
 - Configurar las opciones de, 75
 - Controladores de, 404
 - Crear
 - en DVI, 74, 404
 - en HTML, 417
 - en PDF, 74, 404
 - en PS, 74, 404
 - manualmente, 75, 76
 - Enlace de
 - a la bibliografía, 125
 - a un fichero, 76, 404
 - a un programa, 404
 - a un servidor de ficheros, 404
 - a una dirección electrónica, 404
 - a una dirección URL, 404
 - a una página, 76
 - en HTML, 426
 - en listas, 98
 - en listas bibliográficas, 411
 - en índices terminológicos, 411

- hora
Definir el comando, 470
Imprimir la, 470
- HTML, 417
barra de navegación, 430
- Conversión a
Comandos de la, 424
de listas y entornos, 429
de tablas, 430
del índice general, 429
Fichero de configuración para, 424
Modificando comandos de, 428
Opciones de, 420
Procedimiento de, 418
- Crear nuevas páginas, 427
- División en varias páginas, 422
- Extensiones de
MATHML, 420
MOZILLA, 420
XHTML, 420
- Fórmulas en, 429
- Fuentes en, 423
- Incluir código, 425
- Incluir gráficos en, 422, 427
- Incluir hiperenlaces en, 426
- Marcos (frames) en, 422
- Notas a pie de página en, 422
- Procedimiento para convertir de L^AT_EX a, 418
- I**
- idioma(s)
cambiar el activo, 166
catalán, 171
con el paquete web, 434
disponibles para babel, 166
español, 171
como principal, 174
euskeria, 171
gallego, 171
principal para babel, 164
usar distintos en un mismo documento, 164
- imprimir
a dos caras, 61
en apaisado, 226, *v.t.* página(s)
- inclinada, Letra, *v.* letra(s)
- incluir
anotaciones en PDF, 412
gráficos, 335
en HTML, 422, 427
hiperenlaces, *v.* hipertexto
objetos en gráficos PSTricks, 347
películas en PDF, 416
- incluir (cont.)
sonidos en PDF, 416
- índice
alfabético, *v.* índice(s) terminológico(s)
de cuadros, 118
Incluir manualmente una entrada en el, 206
Título del, *v.* título
de figuras, 86
Incluir manualmente una entrada en el, 206
Título del, *v.* título
general, 57
en HTML, 428, 429
Formato de las entradas del, 206
Incluir manualmente una entrada en el, 206
Números iniciales del, 207
por capítulos, 272
Título del, *v.* título
Unidades de estructura del, *v.* unidades de estructura
índice(s) terminológico(s)
Acentos en, 237
Alfabetización de, 237
Cabecera de los grupos alfabéticos en, 242
Caracteres reservados en, 236
compilación a trozos, 246
Generación con MAKEINDEX de, 233
modificar el fichero de estilo, 241
opciones, 240
Hiperenlaces en, *v.* hipertexto
Intervalos de páginas en, 236
Mayúsculas y minúsculas en, 237
Personalizar el número de página en, 236
Referencias internas en
véase, 235
véase también, 235
Símbolos en, 237
Títulos de, *v.* título
Varios, 246
informe técnico, 49
instalación de L^AT_EX, 10–11
integrales, *v.* fórmulas
interlínea, 29, 162
introducción, 54
invertir una cadena de texto, 487
itálica, *v.* letra(s)
Corrección, 43
- ítem, *v.* listas
- J**
- justificar, *v.t.* texto
horizontalmente, 26, 153, 318
verticalmente, 153, 163

L

- \LaTeX*, Logotipo de, 13
- \LaTeX* a HTML, *v.* HTML
- letra(s), *v.t.* fuente(s)
 - de máquina de escribir (typewriter), 42
 - enfatizada, 44
 - griegas en fórmulas, 134
 - inclinada, 42
 - italica, 42, 44
 - negrita, 43
 - normal, 42
 - Tamaño de la, 44, 59
 - recta, 42
 - sin adornos (sanserif), 42
 - subrayada, 44
 - Tamaños de, 44
 - versalita, 42
 - voladita, 37, 172
- leyenda(s)
 - Anchura, márgenes y sangría de la, 305
 - Antetítulo y delimitador de la, 307
 - Atributos de los tipos en la, 308
 - Diferentes formatos de, 306
 - en cajas marginadas, 312
 - en cuadros, 118
 - en tablas muy largas, 305
 - Nuevos estilos de, 307
 - Suprimir antetítulo y número en la, 308
- libra esterlina (£), 38
- libros, 49
 - División de los, 56
- ligaduras, 178
- línea(s), *v.t.* filete(s)
 - Anchura de las, *v.* página(s)
 - blancas, 23, *v.t.* párrafos, Crear
 - de objetos repetidos, 471
 - huérfanas, 67
 - Iniciar nueva, 23
 - que no se cortan bien, 25, 230
 - Margen de tolerancia en, 230
 - Respetar las, 202
 - Salto de, 23
 - Separación de, *v.* interlínea
 - viudas, 67
- listas
 - con viñetas, 97
 - descriptivas, 98
 - en HTML, 429
 - enmarcadas, 328
 - Etiquetas en
 - con viñetas, 280

listas (cont.)

- descriptivas, 280
- generales, 285
- numeradas, 277
- generales, 281
- ítems, 97
- numeradas, 97
- Parámetros de las, 281
- por etapas, 459
- Referencias cruzadas en, 278
- Representación del contador en, 277
- Símbolos en viñetas de las, 191
- llaves, *v.* fórmulas

logotipo

- de una presentación, 456
- \LaTeX*, 13
- T\EX*, 13, 321

longitud(es)

- Aumentar el valor de una, 469
- Comparación de, 477
- Conocer el valor de una, 143
- Crear nuevas, 196, 468
- de un objeto, 197
- Disminuir el valor de una, 469
- Dividir una, 470
- elástica, 142, 196, 197, 468
- Manipulación global de, 466
- Modificar el valor de una, 143
- Modificar una, 196
- Multiplicar una, 470
- rígida, 142, 196, 468
- Separador decimal de, 139
- Unidades de
 - elásticas, 198
 - rígidas, 143

M

- mancha, *v.* página(s)
- máquina de escribir, *v.* letra(s)
- marca al agua, *v.* filigrana
- marca registrada (®), 38
- marcas
 - en el pie de las páginas, 107
 - filetes, 220
 - selectores de posición, 218
- en la cabecera de las páginas, 51, 215
 - Anchura de las, 218
 - filetes, 220
 - Más, 223
 - Problemas de determinación de, 51, 213
 - selectores de posición, 218
- en las listas, *v.* listas, Etiquetas en

- marcas (cont.)
en las unidades con estrella, 216
Modificar
 del estilo `fancy`, 221
 del estilo `headings`, 214
 del estilo `myheadings`, 216
 en cualquier estilo, 221
Páginas sin, 64, 216
marco, *v.* cajas
margen(es)
 de impresión
 horizontal, 65
 vertical, 67
 de páginas, 64
 a derecha, 65
 a izquierda, 67
 para un párrafo, 201
MATHML, *v.* HTML
matrices, *v.* fórmulas
mayúsculas y minúsculas, 214
 en las cabeceras, 221
modo de trabajo
 horizontal, 315, 478
 matemático, 316, 478
 Acciones al iniciar el, 475
 ordinario, 127, 358
 resaltado, 127, 358
 vertical, 316, 478
MOZILLA, *v.* HTML
multiplicar
 un contador, 470
 una longitud, 470
- N**
- negrita, *v.* letra(s)
nodos, conexión de objetos, 350
nombre(s)
 de funciones
 Acentuar, 391
 Definir nuevos, 390
 en español, 391
 Espacios en, 391
 de un comando, 90
notación hexadecimal, 487
notas
 a pie de página, 105
 Contador de, 105
 en `minipage`, 149
 en `twocolumn`, 106
 en el título del documento, 53, 106
 en HTML, 422
 en tablas, 116
notas (cont.)
 Llamadas múltiples a, 107
 Marcas de separación para, 224
 Marcas especiales en, 108
 Número de, 107
 Parámetros de las, 224
 Separación del texto en, 226
 Separación entre, 224
 Texto de, 107
 al margen, 108
 Parámetros de las, 225
numeración
 de las fórmulas, *v.* fórmulas, numeradas
 de unidades de estructura, *v.* unidades de estructura
número(s)
 combinatorios, *v.* fórmulas
 Comparación de, 477
 referenciar el de una página, 72
 en estilo antiguo, 130
- O**
- objeto(s)
 enmarcados, *v.* cajas
 flotantes, *v.* entornos flotantes
 Iniciar página con, 304
 Reflejar un, 339
 Repetir un, 471
 Rotar un, 338
 ocultar texto, 485, 486
opciones
 de la clase de documento, 58
 Disponibilidad de, 59
 de los paquetes, 16
 para visualizar PDF, 407
ordinales, 37
organismo del autor de la presentación, 456
ortografía, Signos de, *v.* signos ortográficos
- P**
- página(s)
 a izquierda y a derecha, 61
 Alargar una, 69
 Altura de, 67
 Anchura de, 67
 apaisadas, 227
 con DVIPDFM y DVIPDFM, 228
 con PDFLATEX, 229
 Cabecera de las, 63
 Contador de, 139
 Cuerpo de las, 63
 de códigos, 16, 34, 202
 de derechos, 54

- página(s) (cont.)
- de modo incremental, 446
 - de una etiqueta, 72
 - del título, *v. documento*
 - Enmarcar, 326
 - Estilos de
 - básicos, 211
 - con objetos flotantes, 222
 - Definir, 221
 - mejorados, 217
 - Redefinir, 221
 - HTML, 417
 - Justificación vertical de, 163
 - Mancha de las, 64
 - Marcas en cabecera y pie de, *v. marcas*
 - Modificar la altura de una, 162
 - Márgenes de las, *v. margen(es)*
 - Notas a pie de, *v. notas*
 - Numeración de las, 57
 - pares e impares, 61
 - Parámetros de las, 224
 - altura de la cabecera, 225
 - altura de la mancha, 224
 - altura de la primera línea, 226
 - anchura de la línea en curso, 225
 - anchura de la mancha, 224
 - anchura de las columnas, 224
 - anchura del texto, 224
 - márgenes de impresión, 224
 - notas a pie, 224–226
 - notas al margen, 225
 - representación, 226
 - separación de la cabecera, 225
 - separación del borde superior, 226
 - tamaño del papel, 225
 - pequeñas dentro de otra página, *v. cajas*
 - Pie de las, 63
 - por etapas, 446
 - Salto de, 69, 162
 - sin marcas en cabecera ni pie, 64, 211, 216
 - paginación, Modificar la, 162
 - palabra(s)
 - clave de una presentación, 436
 - División de, 24
 - Algoritmos de, 25
 - Problemas en la, 24
 - Separación de, 33
 - viudas, 34, 161
 - panel de navegación
 - a izquierda o derecha, 441
 - Color del, 439
- panel de navegación (cont.)
- en HTML, 430
 - Fondo del, 439
 - Personalizar el, 441
 - Separación del texto del, 442
- papel, Tamaño(s) de
- Declarar, 225
 - estándar, 60
 - Seleccionar, 60
- paquetes, 16
- parágrafo (\$), Signo de, 38
- parámetros
- de las listas, *v. listas*
 - de las páginas, *v. página(s)*
 - de las tablas, *v. tablas*
 - de los entornos flotantes, *v. entornos flotantes*
 - de objetos PSTricks, 353
- párrafo(s)
- Acciones al iniciar un, 475
 - con formas especiales, 473, 474
 - Crear, 23
 - español, en triángulo, 306
 - Modificar número de líneas de un, 161
 - no sangrado, 23
 - sangrado, 473
 - Sangría en, 23
 - Separación de, 23
- partes, 50
- Antetítulo de las, *v. antetítulo*
 - pausa en una presentación, 446
- PDF, *v. salida, Formato de*
- Autor de un documento, 409
 - Incluir en
 - anotaciones, 412
 - películas, 416
 - sonidos, 416
 - Opciones para visualizar, 407
 - Palabras clave de un documento, 409
 - Resumen de un documento, 409
 - Título de un documento, 409
- perfil, *v. fuente(s)*
- pie
- de cuadros, *v. cuadros, Leyenda de*
 - de figuras, *v. figuras, Leyenda de*
 - de página, *v. marcas*
 - Notas a, *v.t. página, Parámetros*
- poemas, 28
 - en formato español, 324
- poesía, *v. poemas*
- porcentaje (%), 32

- porciones de una página, *v.* presentación(es), Subpágina(s) de una
posicionar un objeto, *v.* justificar
PostScript (PS), *v.* salida, Formato de preámbulo, *v.* texto fuente
presentación(es), 433

 - Acumular subpáginas en, 455
 - Animaciones en pantalla de una, 459
 - Asunto de la, 436
 - Autor de la, 436, 456
 - Borrador de la, 455
 - Color del fondo de la, 439
 - Diferentes estilos de, 454, 455
 - Dimensiones de la pantalla en una, 435
 - Dirección electrónica del autor de la, 436, 456
 - Directorio de la, 437
 - Fondo del panel de una, 439
 - Fuente en una
 - para el contenido, 458
 - para el título, 458
 - para las leyendas, 458
 - Incluir en una
 - los botones de navegación, 442
 - objetos en el fondo, 439
 - Logotipo del organismo en una, 456
 - Modificar el color del panel de la, 439
 - Modificar la filigrana de una, 439
 - Márgenes de separación en una, 435
 - Número de versión de la, 436
 - Organismo del autor de la, 456
 - Páginas titulares de una, 458
 - Palabras clave de la, 436
 - Parámetros de la, 436
 - Página del título de una, 436
 - Realizar pausas en una, 446
 - salida
 - PDF, 455
 - PS, 455
 - en blanco y negro, 455
 - en color, 455
 - Seleccionar etapas en una, 460
 - Subpágina(s) de una, 449
 - Asignar niveles a una, 450
 - Enlazar con la primera, 449
 - Mostrar en modo incremental las, 452
 - Mostrar en modo resaltado las, 451
 - Subtítulo de la, 456
 - Título de la, 436, 456
 - transiciones
 - en prosper, 457
 - entre páginas, 446

resumen, 54

Título del, *v.* título

rotar

gráficos, 337

un objeto, 338

una figura o cuadro, 340

una leyenda, 340

S

salida

en blanco y negro, 455

en color, 455

Formato de, *v.t.* fichero(s)

DVI, 4

HTML, 417

PDF, 7, 455

PostScript (PS), 6, 455

salto(s)

de línea, *v.* líneas

de línea en fórmulas, *v.* fórmulas

de página, 69

en fórmulas, 388

sangría, *v.* párrafos, 473

sanserif, *v.* letra(s)

secciones, 50

separación entre operadores matemáticos, 359

separar, *v.* longitud(es)

series, *v.* fuente(s)

signo(s)

calderón (¶), 38, 39

comillas(« ‘ ’ »), 35

copyright (©), 38

dólar (\$), 32

euro (€), 39

libra esterlina (£), 38

marca registrada (®), 38

matemáticos, *v.* fórmulas

ortográficos, 34–39

disponibles en el teclado, 35

para indicar un espacio (␣), 38

para los grados (°), 38

para ordinales, 37

parágrafo (§), 38

porcentual (%), 32

Espacio antes del, 33

tilde, 32

símbolos

matemáticos, 359

delimitadores de apertura, 360, 367

delimitadores de cierre, 360, 367

Negación de, 360

operadores binarios, 360, 363

símbolos (cont.)

operadores de relación, 360, 364–366

operadores de tamaño variable, 360, 393

ordinarios, 134, 360–362

Redefinir, 361

signos de puntuación, 360, 370

químicos, 135

Definir nuevos, 137

índice terminológico, *v.* índices terminológicos(s)

sin adornos, *v.* letra(s)

sintaxis de LATEX

argumentos, 13

caracteres reservados, 31

comandos, 13

entornos, 13

sistematización de tareas

al iniciar cajas, 476

al iniciar fórmulas, 475

al iniciar párrafos, 475

sombra de cajas, *v.* cajas, 354

subpágina de una presentación, *v.* presentación(es)

subrayar, 44

subtítulo de una presentación, 456

subíndices, *v.* fórmulas

sumar

a un contador, 469

a una longitud, 469

sumatorios, *v.* fórmulas

superíndice, *v.* fórmulas

T

Tabla Periódica de elementos químicos, 137

tablas

Anchura de columnas en, 112

Celdas en, 111

Columnas en, 112

con color de fondo

en celdas, 299

en columnas, 298

en filas, 299

con filas agrupadas, 295

con filetes, 113

de color, 300

dobles, 296

gruesos usando array, 291

con varias leyendas, 291

divididas entre páginas, 291

en HTML, 430

Especificadores de, 112

Filas en, 111

marginadas, *v.* cajas

muy largas, 291

- tablas (cont.)
- Parámetros de las, 288
 - Posición de las, 112
 - Rayas en, 113
 - Separadores en, 112
 - Sintaxis de las, 111
 - tipos de columna, 112
 - estándar, 290
 - nuevos, 290
 - tamaño de los tipos, *v. fuente(s)*
 - taquigrafía
 - apóstrofo ('), 171
 - Métodos de
 - Activar, 170
 - Definir, 170
 - Desactivar, 170, 176
 - Niveles de, 170
 - para catalán, 172
 - para español, 172
 - para euskera, 173
 - para gallego, 173
 - teorema(s)
 - con el paquete *amsthm*, 398
 - Demostración de, 400
 - Entornos tipo, 396
 - Estilo de, 398
 - Intercambiar etiqueta y leyenda en, 399
 - TeX, Logotipo de, 13, 321
 - texto(s)
 - alineado a derecha o a izquierda, 26
 - Anchura del, *v. página(s)*
 - bilingües, 103
 - centrado, 25
 - con sangrías, 28
 - en bandera por la izquierda o derecha, 26
 - en color, *v. color(es)*
 - en paralelo, 103
 - enmarcado, *v. cajas*, 148
 - Espaciado del, *v. interlínea*
 - fuente, 3
 - Comentarios en el, 32
 - Cuerpo del, 13
 - Preámbulo del, 13, 16
 - Saltos de línea en el, 22
 - Invertir una cadena de, 487
 - Justificar, 25
 - tilde (~), 32
 - tipos, *v. fuente(s)*
 - título, *v.t.* antetítulo
 - de la bibliografía, 250, 268
 - de la presentación, 436, 456
 - título (cont.)
 - de un documento, *v. documento*
 - PDF, 409
 - del glosario, 249
 - del resumen, 56
 - del índice
 - de cuadros, 205
 - de figuras, 205
 - general, 57, 205
 - terminológico, 234, 247
 - transiciones
 - Archivo de, 446
 - en prosper, 457
 - entre páginas originales, 446
 - entre subpáginas, 447
 - transparencia(s), 49
 - de una presentación, *v. presentación(es)*
 - Número de etapas en, 459
 - tríptico, 229
 - typewriter, *v. letra(s)*
- U**
- unidades de estructura
 - Antetítulos de, *v. antetítulo*
 - con asterisco, 52
 - en el índice general, 51, 58
 - Formato de la numeración de las, 204
 - Jerarquía de las, 50
 - listadas en el índice general, 204
 - Nivel de las, 203
 - Numeración de las, 51
 - que se numeran, 203
 - URL, Crear hiperenlaces a, 404
- V**
- variables lógicas, *v. condicionales*
 - versalita, *v. letra(s)*
 - versión
 - borrador
 - de una presentación, *v. presentación(es)*
 - del documento, 60
 - de la presentación, 436
 - final
 - de una presentación, *v. presentación(es)*
 - del documento, 60
 - versos, *v. poemas*
 - «viene de la página anterior...», 223
 - viñetas, *v. listas*
 - virgulilla (~), *v. tilde*
 - visionar la compilación, 5

visores de ficheros

DVI, 5, 81

PDF, 8

PS, 7

voladita, *v.* letra(s)

X

XHTML, *v.* HTML

Índice terminológico

Símbolos

!, 370
\!, 134
#, 32
\# (#), 32, 362
\$, 32, 127, 358
\$\$, 358
\\$ (\$), 32
%, 32
\% (%), 32, 175
\&, 32
\& (&), 32, 362
, 36
' , 36
-, 36, 363
\-, 24, 173
--, 36
---, 36
--, 36
~, 36
^, 36
\^, 32, 130
~, 32
_, 32, 130
~, 32
\~, 32
(, 367
\(, 127, 358
\), 127, 358
[, 367
\[, 128, 358
], 367
\], 128, 358
\{, 32
\}, 31

\}, 32
*, 113
*, 363
+, 363
\,, 134
\..., 37, 175
\/, 370
\:, 134
\;, 134
\<, 364
\<<, 36
\=, 364
\>, 364
\>>, 36
\?, 370
\@ (@), 160
\, 31
\\", 23
_, 68
\|, 370

A

a4paper, opción, 60
a5paper, opción, 60
\abovecaptionskip, longitud, 303
\abovedisplayshortskip, longitud, 400
\abovedisplayskip, longitud, 400
abstract, entorno, 54
\abstractname, 171
\accentedoperators, 391
accents, paquete, 369
accumulate, opción, 455
achicago, paquete, 266
ACROBAT, ejecutable, 8
ACROBAT READER, ejecutable, 8, 10
\Acrobatmenu, 411

\activatequoting, 177
activeacute, opción, 171
activegrave, opción, 171
\acute (\r), 368
\addcontentsline, 206
\addto, 169
\addtocontents, 207
\addtocounter, 141, 466
\addtolength, 143, 466
\AddToPanelTemplate, 439
\AddToTemplate, 439
\advance, 469
\AE, 38
\ae, 38
afterpag, paquete, 304
\afterpage, 304
\aleph (\aleph), 361
alien, opción, 454
alienglow, opción, 455
align, entorno, 385
alignat, entorno, 385
aligned, entorno, 386
alignedat, entorno, 386
\allowdisplaybreaks, 388
\Alph, 140, 174
\alph, 140, 174
\alpha (\alpha), 134
\alsoname, 171, 236
\amalg (II), 363
amsart, clase, 402
amsbook, clase, 402
amsbsy, paquete, 357
amscd, paquete, 395
amsfonts, paquete, 357
amsmath, paquete, 126, 357
amsopn, paquete, 357
amsref, paquete, 402

- amssymb, paquete, 357, 434, 452
 amstex, paquete, 136, 177
 amstext, paquete, 357
 amsthm, paquete, 398
 anchorcolor, opción, 406
 $\backslash and$, 53
 and, 260
 $\angle (\angle)$, 362
 angle, opción, 337
 \appendix , 56
 \appendixname , 171
 $\approx (\approx)$, 364
 $\approxeq (\approxeq)$, 364
 \arabic , 140
 $\arccos (\arccos)$, 389
 $\arcsen (\arcsen)$, 390
 $\arcsin (\arcsin)$, 389
 $\arctan (\arctan)$, 389
 $\arctg (\arctg)$, 390
 $\arg (\arg)$, 389
 array, entorno, 378
 array, paquete, 289, 378
 \arraycolsep , longitud, 288
 \arrayrulecolor , 300
 \arrayrulewidth , longitud, 288
 \arraystretch , 288
 $\Arrowvert (\|)$, 367
 $\arrowvert (\|)$, 367
 article, clase, 49
 $\ast (\ast)$, 363
 $\asymp (\asymp)$, 364
 \author , 53, 436, 456
 autumn, opción, 454, 455
 AUX, fichero, 5, 246
 avant, paquete, 187
 azure, opción, 454, 455
- B**
- b5paper , opción, 60
 babel, paquete, 17, 130, 164, 391, 435
 $\backepsilon (\backepsilon)$, 366
 background, paquete, 445, 447
 \backmatter , 56
 $\backprime (\backprime)$, 362
 backref, opción, 411
 $\backsimeq (\backsimeq)$, 364
 $\backsimeqeq (\backsimeqeq)$, 364
 \backslash , 367
 \bar{x} , 368
 $\barwedge (\barwedge)$, 363
 \barwedge , longitud, 162
 \baselineskip , longitud, 29
- baseurl, opción, 411
 \mathbf{basque} , opción, 166, 171
 \mathbf{bb} , opción, 336
 $\mathbf{Bbbk} (\mathbb{k})$, 361
 BBL, fichero, 253
 $\mathbf{Bcenter}$, entorno, 328
 $\mathbf{Bdescription}$, entorno, 328
 $\mathbf{because} (\because)$, 366
 \mathbf{begin} , 13
 $\mathbf{begin\{document\}}$, 13, 16
 $\mathbf{begingroup}$, 466
 $\mathbf{belowcaptionskip}$, longitud, 303
 $\mathbf{belowdisplayshortskip}$, longitud, 400
 $\mathbf{belowdisplayskip}$, longitud, 400
 $\mathbf{Benumerate}$, entorno, 328
 $\mathbf{Beqnarray}$, entorno, 328
 $\mathbf{Beqnarray*}$, entorno, 328
 $\mathbf{\beta}$, 134
 $\mathbf{\beth} (\beth)$, 361
 $\mathbf{between} (\langle\rangle)$, 366
 $\mathbf{Bflushleft}$, entorno, 328
 $\mathbf{Bflushright}$, entorno, 328
 $\mathbf{bfseries} (\mathbf{A})$, 43
 \mathbf{bgadd} , 448
 \mathbf{bgadd} , opción, 448
 $\mathbf{bgaddcenter}$, 448
 $\mathbf{bgclear}$, 448
 \mathbf{bgroup} , 325, 466
 $\mathbf{bibindent}$, 255
 $\mathbf{bibitem}$, 122
 $\mathbf{bibliography}$, 252
 $\mathbf{bibliography*}$, 269
 $\mathbf{bibliographystyle}$, 252
 $\mathbf{bibliographystyle*}$, 269
 $\mathbf{bibliographyunit}$, 269
 $\mathbf{bibname}$, 171, 250
 $\mathbf{bibunit}$, entorno, 269
 $\mathbf{bibunits}$, paquete, 268
 $\mathbf{bigcap} (\bigcap)$, 393
 $\mathbf{bigcirc} (\bigcirc)$, 363
 $\mathbf{bigcup} (\bigcup)$, 393
 \mathbf{Biggl} , 377
 \mathbf{biggl} , 377
 \mathbf{Biggr} , 377
 \mathbf{biggr} , 377
 \mathbf{Bigl} , 377
 \mathbf{bigl} , 377
 $\mathbf{bigdot} (\bigodot)$, 393
 $\mathbf{bigoplus} (\bigoplus)$, 393
 $\mathbf{bigotimes} (\bigotimes)$, 393
 \mathbf{Bigr} , 377
 \mathbf{bigr} , 377
 $\mathbf{bigskip}$, 68
 $\mathbf{bigskipamount}$, longitud, 142
 $\mathbf{bigsqcup} (\bigcup)$, 393
 $\mathbf{bigstar} (\bigstar)$, 362
 $\mathbf{bigtriangledown} (\bigtriangledown)$, 363
 $\mathbf{bigtriangleup} (\bigtriangleup)$, 363
 $\mathbf{biguplus} (\biguplus)$, 393
 $\mathbf{bigvee} (\bigvee)$, 393
 $\mathbf{bigwedge} (\bigwedge)$, 393
 \mathbf{binom} , 133
 $\mathbf{Bitemize}$, entorno, 328
 $\mathbf{blacklozenge} (\blacklozenge)$, 362
 $\mathbf{blacksquare} (\blacksquare)$, 362
 $\mathbf{blacktriangle} (\blacktriangle)$, 362
 $\mathbf{blacktriangledown} (\blacktriangledown)$, 362
 $\mathbf{blacktriangleleft} (\blacktriangleleft)$, 366
 $\mathbf{blacktriangleright} (\blacktriangleright)$, 366
 BLG, fichero, 253
 \mathbf{Blist} , entorno, 328
 $\mathbf{Bmatrix}$, entorno, 379
 $\mathbf{bmatrix}$, entorno, 379
 \mathbf{bmod} , 390
 $\mathbf{boldsymbol}$, 366, 367
 book, clase, 49
 bookman, paquete, 187
 bookmarks, opción, 407
 $\mathbf{bookmarksnumbered}$, opción, 407
 $\mathbf{bookmarksopen}$, opción, 407
 $\mathbf{bookmarksopenlevel}$, opción, 407
 $\mathbf{bot} (\bot)$, 362
 $\mathbf{botfigrule}$, 304
 $\mathbf{bottomfraction}$, 302
 $\mathbf{bottomnumber}$, contador, 302
 $\mathbf{bowtie} (\bowtie)$, 366
 \mathbf{box} , 323
 $\mathbf{boxdot} (\boxdot)$, 363
 \mathbf{boxed} , 371
 $\mathbf{boxlength}$, longitud, 311
 $\mathbf{boxmaxdepth}$, 319
 $\mathbf{boxminus} (\boxminus)$, 363
 $\mathbf{boxplus} (\boxplus)$, 363
 \mathbf{boxput} , 327
 $\mathbf{boxput*}$, 327
 $\mathbf{boxtimes} (\boxtimes)$, 363
 bp, unidad de longitud, 143
 $\mathbf{bracevert} (\,\,\,)$, 367
 breaklinks, opción, 405
 $\mathbf{breve} (\breve{x})$, 368
 $\mathbf{buildpanel}$, 441

- \bullet (•), 363
 \Bumpeq (≈), 364
 \bumpeq (≂), 364
- C**
 calc, paquete, 487
 calrsfs, paquete, 365
 \Cap (⊸), 363
 \cap (∩), 363
 \caption, 84, 86, 293
 \caption*, 293
 caption2, paquete, 305
 \captionfont, 308
 \captionindent, longitud, 307
 \captionlabel, 307
 \captionlabeldelim, 307
 \captionlabelfalse, 308
 \captionlabelfont, 308
 \captionlabelsep, 307
 \captionlabeltrue, 308
 \captionlinewidth, longitud, 307
 \captionmargin, longitud, 306
 \captionsIdioma, 169
 \captionstyle, 307
 \captiontext, 307
 \captionwidth, longitud, 306
 @captcha, 313
 cases, entorno, 387
 catalan, opción, 165, 166, 171
 cc, unidad de longitud, 143
 CD, entorno, 395
 \cdot (·), 363
 \cdots, 130
 \cdots (…), 370
 center, entorno, 25
 \centerdot (.), 363
 \centering, 27
 \centerline, 26, 318
 centertags, opción, 358
 \cfrac, 375
 chancery, paquete, 187
 chapter, contador, 139, 204
 \chapter, 50
 \chapter*, 50
 \chaptermark, 51
 \chaptername, 171, 205
 \char, 189
 \check (✗), 368
 chemsym, paquete, 135
 \chi (χ), 134
 \circ (○), 363
 \circeq (≓), 364
- \circlearrowleft (○), 365
 \circlearrowright (○), 365
 \circledast (⊛), 363
 \circledcirc (○), 363
 \circleddash (○), 363
 \circledS (Ⓢ), 361
 \cite, 123, 252
 cite, paquete, 266
 \cite*, 270
 citebordercolor, opción, 406
 citecolor, opción, 406
 \citen, 266
 citesort, paquete, 266
 clases
 amsart, 402
 amsbook, 402
 article, 49
 book, 49
 letter, 49
 proc, 49
 prosper, 452
 report, 49
 slides, 49
 \cleaders, 472
 \ClearAllTemplates, 440
 \ClearBuildPanel, 444
 \cleardoublepage, 69, 481
 \clearpage, 69, 102, 163, 305
 \ClearPanelTemplate, 440
 \ClearTextTemplate, 440
 \cline, 117
 CLO, fichero, 59
 CLS, fichero, 49
 \clubpenalty, 67
 \clubsuit (♣), 362
 cm, unidad de longitud, 143
 cm-super, paquete, 17
 \colon (:), 370
 \color, 46, 275
 color, paquete, 45, 273, 434
 colorBG, opción, 455
 \colorbox, 275
 \ColorFoot, 458
 colorlinks, opción, 75, 406
 colortbl, paquete, 298
 \columnbreak, 103
 \columncolor, 298
 \columnsep, longitud, 102
 \columnseprule, longitud, 102
 \columnwidth, longitud, 224
 command, opción, 337
- \complement (⊸), 361
 comprehensive, paquete, 192
 \Configure, 428
 \ConfigureEnv, 429
 \ConfigureList, 429
 \cong (≡), 364
 contadores
 bottomnumber, 302
 chapter, 139, 204
 dbltopnumber, 302
 enumi, 277
 enumii, 277
 enumiii, 277
 enumiv, 277
 equation, 128
 figure, 85
 MaxMatrixCols, 379
 mpfootnote, 150
 page, 139
 parrafo, 475
 part, 205
 secnumdepth, 204
 section, 139, 204
 subsection, 205
 subsubsection, 205
 table, 118
 tocdepth, 204
 topnumber, 302
 totalnumber, 302
 contemporain, opción, 454, 455
 \contentsname, 171, 205
 \coprod (⊔), 393
 \copy, 323
 \copyright (©), 38
 \cos (cos), 389
 \cosec (cosec), 391
 \cosh (cosh), 389
 \cot (cot), 389
 \cotg (cotg), 391
 \coth (coth), 389
 courier, paquete, 187
 \CrearLista, 485
 \csc (csc), 389
 \csname, 479
 \Css, 425
 \Cup (⊸), 363
 \cup (∪), 363
 \curlyeqprec (≲), 364
 \curlyeqsucc (≳), 364
 \curlyvee (⊲), 363
 \curlywedge (∧), 363

\curvearrowleft (\curvearrowleft), 365
 \curvearrowright (\curvearrowright), 365

D
 \dag (\dag), 38
 \dagger (\dagger), 363
 \daleth (\daleth), 361
 dansk, opción, 434
 darkblue, opción, 454, 455
 \dashlength, longitud, 311
 \dashv (\dashv), 366
 \datatplot, 344
 \date, 53
 \dateIdioma, 169
 \dbinom, 133
 \dblfigrule, 304
 \dblfloatpagefraction, 302
 \dblfloatsep, longitud, 303
 \dbltextfloatsep, longitud, 303
 \dbltopfraction, 302
 dbltopnumber, contador, 302
 dcolumn, paquete, 120, 328
 dd, unidad de longitud, 143
 \ddag (\ddag), 38
 \ddagger (\ddagger), 363
 \dddot ($\ddot{\cdot}$), 368
 \ddotdot ($\ddot{\cdot}$), 368
 \ddot ($\ddot{\cdot}$), 368
 \ddots (\ddots), 370
 \deactivatequoting, 177, 396
 \deactivatetilden, 177
 \decimalcomma, 392
 \decimalpoint, 392
 \DeclareInputText, 39
 \DeclareMathOperator, 390
 \DeclareMathOperator*, 390
 \DeclareMathSymbol, 361, 402
 \DeclareSymbolFont, 361, 402
 \def, 463, 464
 \DefaultTransition, 456
 \definecolor, 274, 342
 \defineshorthand, 170
 \deg (deg), 389
 \degrees, 342
 delarray, paquete, 379
 \Delta (Δ), 134
 \delta (δ), 134
 description, entorno, 98
 \descriptionlabel, 280
 designi, opción, 435
 designii, opción, 435

designiii, opción, 435
 designiv, opción, 435
 designv, opción, 435
 \det (det), 389
 \dfrac, 132
 \diagdown (\diagdown), 362
 diagram, paquete, 395
 \diagup (\diagup), 362
 \diamond (\diamond), 363
 \diamondsuit (\diamondsuit), 362
 \digamma (\digamma), 134
 \dim (dim), 389
 dimen, 196
 \ding, 191
 dingautolist, entorno, 191
 \dingfill, 191
 \dingline, 191
 dinglist, entorno, 191
 \disableindex, 248
 \displaybreaks, 388
 \displaylimits, 393
 displaymath, entorno, 128, 358
 \displaystyle, 132, 371
 \displayVersion, 456
 \Dissolve, 447
 \div (\div), 363
 \divideontimes (*), 363
 djslib, paquete, 461
 \documentclass, 49
 \dot ($\dot{\cdot}$), 368
 \doteq (\doteq), 364
 \doteqdot (\doteqdot), 364
 \dotfill, 135, 153
 \dotlessi, 391
 \dotplus ($\dot{+}$), 363
 \dots (...), 37, 370
 \dotsb (...), 370
 \dotsc (...), 370
 \dotsi (...), 370
 \dotsm (...), 370
 \dotso (...), 370
 \doublebarwedge (\barwedge), 363
 \doublebox, 148
 \doublerulesep, longitud, 289
 \doublerulesepcolor, 300
 doublespace, paquete, 29
 \Downarrow (\Downarrow), 365
 \downarrow (\downarrow), 365
 \downbracefill, 199
 \downdownarrows (\downdownarrows), 365
 \downharpoonleft (\downharpoonleft), 365

\downharpoonright (\downharpoonright), 365
 \dp, 323
 draft, opción, 60, 75, 335, 455, 508, 513
 dutch, opción, 434
 DVI, fichero, 5
 dvipdf, opción, 274
 DVIPDFM, ejecutable, 8, 15, 81
 dvipdfm, opción, 274, 434, 445
 DVIPS, ejecutable, 7, 15, 81, 179
 dvips, opción, 45, 74, 274, 434
 dvipsnames, opción, 46
 dvipsone, opción, 274, 434
 DVIWIN, ejecutable, 81
 dviwin, opción, 274
 DVIWINDO, ejecutable, 81
 dviwindo, opción, 274, 434

E
 EBB, ejecutable, 81
 \edef, 467
 eepic, paquete, 356
 \egroup, 325, 466
 ejecutables
 ACROBAT, 8
 ACROBAT READER, 8, 10
 DVIPDFM, 8, 15, 81
 DVIPS, 7, 15, 81, 179
 DVIWIN, 81
 DVIWINDO, 81
 EBB, 81
 EPS2PDF, 334
 EPSTOPDF, 334
 GFTOPK, 178
 GHOSTSCRIPT, 7, 11, 186
 GHOSTVIEW, 7, 11
 GSVIEW, 7, 11, 334
 JIBTEXMANAGER, 265
 MAKEINDEX, 233, 240
 PLTOTF, 178
 PS2PDF, 334
 PST2PDF, 335
 T4HT, 418
 TEX4HT, 418
 TFTOPL, 178
 TKBIBTEX, 265
 WINBIBDB, 265
 XDVI, 179
 XINDY, 238
 YAP, 81, 179
 \ell (ℓ), 361
 \else, 476

- \em, 44
\em, unidad de longitud, 143
\email, 436, 456
\emph, 44
empty, estilo de página, 211
\emptyset (), 362
emtex, opción, 274
\end, 13
\endcsname, 479
\end{document}, 13, 16
\endfirsthead, 292
\endfoot, 292
\endgroup, 466
\endhead, 292
\EndHPage{}, 428
\endlastfoot, 292
endnotes, paquete, 329
\EndPreamble, 424
\enlargethispage, 69
\enlargethispage*, 69
\enskip, 68
\enspace, 68
\ensuremath, 464, 478
entornos
 abstract, 54
 align, 385
 alignat, 385
 aligned, 386
 alignedat, 386
 array, 378
 Bcenter, 328
 Bdescription, 328
 Benumerate, 328
 Beqnarray, 328
 Beqnarray*, 328
 Bflushleft, 328
 Bflushright, 328
 bibunit, 269
 Bitemize, 328
 Blist, 328
 Bmatrix, 379
 bmatrix, 379
 cases, 387
 CD, 395
 center, 25
 description, 98
 dingautolist, 191
 dinglist, 191
 displaymath, 128, 358
 enumerate, 97, 174
 eqnarray, 381
 equation, 128, 358
 equation*, 129, 358
 figure, 84
 figure*, 86
 flalign, 385
 flushleft, 26
 flushright, 26
 gather, 385
 gathered, 386
 hyphenrules, 168
 Itemize, 459
 itemize, 97, 174
 itemstep, 459
 list, 281
 longtable, 305
 lrbox, 153
 math, 127, 358
 matrix, 379
 minipage, 149
 multicolpar, 103
 multicols, 102
 multiline, 384
 otherlanguage, 167
 otherlanguage*, 167
 Piautolist, 191
 picture, 148, 355
 Pilist, 191
 pmatrix, 379
 poema, 324
 proof, 399
 psmatrix, 352
 pspicture, 341
 quotation, 27
 quote, 27
 quoting, 175
 shortindexingon, 248
 sidewaysfigure, 340
 sidewaystable, 340
 slide, 453, 457
 sloppypar, 230
 smallmatrix, 380
 split, 358, 383
 subarray, 394
 subequations, 388
 table*, 119
 tabular, 111
 TeXtoEPS, 350
 thebibliography, 123
 thegloss, 271
 theglossary, 249
 theindex, 233
 titlepage, 54
 trivlist, 284
 verbatim, 202
 verse, 28
 Vmatrix, 379
 vmatrix, 379
 enumerate, entorno, 97, 174
 enumi, contador, 277
 enumii, contador, 277
 enumiii, contador, 277
 enumiv, contador, 277
 epic, paquete, 356
 EPS2PDF, ejecutable, 334
 \epsilon, 134
 EPSTOPDF, ejecutable, 334
 epstopdf, paquete, 335
 \eqcirc (=), 364
 eqnarray, entorno, 381
 eqnarray*, entorno, 381
 \eqref, 128
 \eqsim (≈), 364
 \eqslantgtr (>), 364
 \eqslantless (<), 364
 equation, contador, 128
 equation, entorno, 128, 358
 equation*, entorno, 129, 358
 \equiv (≡), 364
 eso-pic, paquete, 434, 448
 \eta, 134
 \etalchar, 264
 \eth (ð), 361
 eucal, paquete, 365
 \EUR, 192
 \EUR (€), 192
 \ euro (€), 39, 192
 \ euro, 192
 eurosym, paquete, 39, 192
 \evensidemargin, longitud, 65, 224
 \everydisplay, 475
 \everyhbox, 476
 \everymath, 475
 \everypar, 475
 everyshi, paquete, 434
 \everyvbox, 476
 ex, unidad de longitud, 143
 executivepaper, opción, 60
 exquiz, paquete, 461
 \exists, 362
 \ExitHPage, 428
 \exp (exp), 389

\expandafter, 479
 ext, opción, 337
 extension, opción, 411
 \extracolsep, 115
 \extramarks, 223
 extramarks, paquete, 223
 \extrarowheight, longitud, 291

F

\fallingdotseq (\leftrightharpoons), 364
 fancy, estilo de página, 217
 fancybox, paquete, 148, 326, 372
 \fancyfoot, 218
 fancyhdr, paquete, 137, 217
 \fancyhead, 218
 \fancyhf, 219
 \fancypage, 326
 \fancypagestyle, 221
 \fancyput, 327
 \fancyput*, 327
 \fbox, 147, 371
 \fboxrule, 147
 \fboxsep, 147
 \fcolorbox, 275
 FD, fichero, 179
 \fi, 476
 ficheros
 AUX, 5, 246
 BBL, 253
 BLG, 253
 CLO, 59
 CLS, 49
 DVI, 5
 FD, 179
 GF, 178
 GLO, 249
 HTML, 417
 IDX, 232
 ILG, 238
 IND, 234
 IST, 241
 LOF, 86, 205
 LOG, 5
 LOT, 118, 205
 LTX, 4
 MF, 178
 PDF, 7, 8
 PFB, 182
 PK, 178
 PL, 178
 PS, 6
 TEX, 4

 TFM, 178
 TOC, 57, 205
 figure, contador, 85
 figure, entorno, 84
 figure*, entorno, 86
 \figurename, 171
 fil, unidad de longitud, 198
 filebordercolor, opción, 406
 filecolor, opción, 406
 \fill, 198
 fill, unidad de longitud, 198
 final, opción, 60, 455
 final|draft, opción, 335
 finnish, opción, 434
 \Finv (\exists), 361
 \firstleftmark, 223
 \firstxmark, 223
 flalign, entorno, 385
 \flat (\flat), 362
 fleqn, opción, 359, 401
 float, paquete, 328
 \floatpagefraction, 302
 \floatsep, longitud, 303
 \flushbottom, 164
 flushleft, entorno, 26
 flushright, entorno, 26
 \fnsymbol, 108, 140, 174
 fontenc, paquete, 17, 180
 \fontencoding, 194
 \fontfamily, 194
 \fontseries, 194
 \fontshape, 194
 \fontsize, 194
 \FontText, 458
 \fontText, 458
 \FontTitle, 458
 \fontTitle, 458
 footbib, paquete, 266
 \footbibliography, 267
 \footbibliographystyle, 267
 \footcite, 267
 \footcite*, 267
 \footnocite, 267
 \footnote, 105
 \footnoterule, 224
 \footnotesep, longitud, 225
 \footnotesize, 44
 \footrule, 220
 \footruleskip, longitud, 220
 \footrulewidth, longitud, 220
 \footskip, longitud, 225

\@for, 484
 \forall (\forall), 362
 forcolorpaper, opción, 444
 \foreignlanguage, 167
 forpaper, opción, 444
 fp, paquete, 488
 \frac, 131
 \frame, 147
 \framebox, 147, 371
 frames, opción, 454, 455
 french, opción, 434
 frenchlinks, opción, 405
 \frenchspacing, 160, 175
 \FromSlide, 460
 \fromSlide, 460
 \fromSlide*, 460
 \frontmatter, 56
 \frown (\frown), 366
 \fussy, 230

G

galician, opción, 166, 171
 \Game (\triangleright), 361
 \Gamma (Γ), 134
 \gamma (γ), 134
 gather, entorno, 385
 gathered, entorno, 386
 \gcd (gcd), 389
 \gdef, 466
 \genfrac, 375
 \geq (\geq), 364
 \geqq (\geqslant), 364
 \geqslant (\geqslant), 364
 german, opción, 434
 GF, fichero, 178
 GFTOPK, ejecutable, 178
 \gg (\gg), 364
 \ggg (\ggg), 364
 GHOSTSCRIPT, ejecutable, 7, 11, 186
 GHOSTVIEW, ejecutable, 7, 11
 \gimel (\beth), 361
 GLO, fichero, 249
 \global, 466
 \gloss, 271
 gloss, paquete, 271
 \glossary, 249
 \glossaryentry, 249
 \glossaryname, 171, 249
 \glossname, 271
 \gnapprox (\approx), 364
 \gneq (\geq), 364
 \gneqq (\geqslant), 364

- \gnsim (\gtrsim), 364
 graphicx, paquete, 80, 273, 335, 355, 452
 \grave{ \grave{x} }, 368
 \greek, 479
 GSVIEW, ejecutable, 7, 11, 334
 \gtrapprox (\gtrapprox), 364
 \gtrdot (\gtreqless), 363
 \gtreqless (\gtreqless), 364
 \gtreqqless (\gtreqqless), 364
 \gtrless (\gtrless), 364
 \gtrsim (\gtrsim), 364
 \gvertneqq (\gvertneqq), 364
 gyom, opción, 454, 455
- H**
 \hangafter, 473
 \hangindent, longitud, 245, 473
 \hat{ \hat{x} }, 368
 \hbar (\hbar), 361
 \HBlinds, 447
 \hbox, 317
 \HChar, 425
 \HCode, 425
 \hdotsfor, 379
 \headheight, longitud, 225
 headings, estilo de página, 211
 \headrule, 220
 \headrulewidth, longitud, 220
 \headsep, longitud, 225
 \headwidth, longitud, 218
 \heartsuit (\heartsuit), 362
 helvet, paquete, 184
 \hexnumber, 478
 \hfil, 199
 \hfill, 153, 199, 398
 \hfilneg, 199
 \hfuzz, longitud, 230
 \hhline, 297, 300
 hhline, paquete, 296
 hidemode, opción, 336
 hidescale, opción, 336
 hiresbb, opción, 336, 337
 \HISplit, 447
 \hline, 114
 \hoffset, 64
 \hoffset, longitud, 65, 224
 \hom (hom), 389
 \hookleftarrow (\hookleftarrow), 365
 \hookrightarrow (\hookrightarrow), 365
 \hora, 471
- \HOSplit, 447
 \HPage, 428
 \hpagecolor, 448
 \phantom, 372
 \hportionwebauthor, 436
 \hportionwebtitle, 436
 \href, 76, 403
 \hrule, 322
 \hrulefill, 153
 \hspace, longitud, 225
 \hslash (\hslash), 361
 \hspace*, 68
 \hspace*, 68
 \hss, 199, 318
 \ht, 323
 HTML, fichero, 417
 \Huge, 44
 \huge, 44
 \hyperdef, 76
 hyperfigures, opción, 411
 hyperindex, opción, 411
 \hyperlink, 75, 449
 \hyperpage, 76
 hyperref, paquete, 74, 273, 434, 452
 \hypersetup, 75
 \hypertarget, 75, 449
 \hyphenation, 25
 hyphenrules, entorno, 168
- I**
 \IBox, 447
 \idotsint (\cdots), 394
 IDX, fichero, 232
 \@idxitem, 245
 \if, 476
 \ifbotfloat, 222
 \ifcase, 478
 \ifdim, 477
 \ifeqnsw, 481
 \iffloatpage, 222
 \ifhmode, 478
 \iflanguage, 168
 \ifmmode, 478
 \@ifnextchar, 481
 \ifnum, 477
 \ifodd, 477
 ifthen, paquete, 177
 \iftopfloat, 222
 \iftwocolumn, 481
 \iftwoside, 481
 \@ifundefined, 481
 \ifvmode, 478
- \ifx, 479
 ignore, opción, 445
 \ignorespaces, 202
 \iiiint (\iiii), 394
 \iiint (\iiii), 394
 \iint (\ii), 394
 ILG, fichero, 238
 \Im (\Im), 361
 \imath (\imath), 369
 \in (\in), 366
 in, unidad de longitud, 143
 \include, 210
 \includegraphics, 82, 336
 \includegraphics*, 82
 \includeonly, 210
 IND, fichero, 234
 \index, 232
 index, paquete, 246
 \indexentry, 232
 \indexname, 171, 234
 \indexproofstyle, 248
 \indexspace, 245
 \inf (\inf), 389
 \infty (∞), 362
 \InitLayout, 442
 \injlim (injlim), 389
 \input, 209
 inputenc, paquete, 16, 202
 insdls, paquete, 461
 \insNavBar, 441
 \insNavBarOff, 441
 \institution, 456
 \int (\int), 393
 \intercal (\intercal), 363
 \intertext, 387
 \intextsep, longitud, 303
 intlimits, opción, 359
 \iota (ι), 134
 IST, fichero, 241
 italic, opción, 434
 \item, 233
 \itemindent, longitud, 283
 Itemize, entorno, 459
 itemize, entorno, 97, 174
 \itemsep, longitud, 283
 itemstep, entorno, 459
 iTexMac, 11
 \upshape (A), 42
- J**
 JBIBTEXMANAGER, ejecutable, 265
 \jmath (\jmath), 369

\jot, longitud, 401

K

\kappa (κ), 134
\kemtkn, 137
\ker (ker), 389
\kern, 321
\keywords, 436
kuvio, paquete, 395

L

\L, 173
\L, 173
\label, 72, 128, 196
\labelenumi, 277
\labelenumii, 277
\labelenumiii, 277
\labelenumiv, 277
\labelitemi, 280
\labelitemii, 280
\labelitemiii, 280
\labelitemiv, 280
\labelsep, longitud, 283
\labelwidth, longitud, 283
\Lambda (Λ), 134
\lambda (λ), 134
landscape, opción, 227
\angle (), 367
\languagename, 168
\languageshorthands, 170
\LARGE, 44
\Large, 44
\large, 44
\lastbox, 324
lastpage, paquete, 222
\lastrightmark, 223
\lastxmark, 223
\LaTeX, 13
latextoc, opción, 437
latin1, opción, 16, 34
\layout, 226
layout, paquete, 226
\layoutspanish, 174
\brace {}, 367
\ceil [], 367
\dots ..., 370
\leaders, 471
\leavevmode, 200, 317
\left, 376
\left., 376
\Leftarrow (\Leftarrow), 365
\leftarrow (\leftarrow), 365

\leftarrowfill, 199
\leftarrowtail (\leftarrowtail), 365
\leftharpoondown (\leftharpoondown), 365
\leftharpoonup (\leftharpoonup), 365
\leftleftarrows (\leftleftarrows), 365
\leftline, 26, 318
\leftmargin, longitud, 283
\leftmargini, longitud, 283
\leftmarginii, longitud, 283
\leftmarginvi, longitud, 283
\leftmark, 212
leftpanel, opción, 441
\Leftrightarrow (\Leftrightarrow), 365
\Leftrightarrow (\Leftrightarrow), 365
\Leftrightarrows (\Leftrightarrows), 365
\leftrightharpoons (\leftrightharpoons), 365
\leftrightsquigarrow (\leftrightsquigarrow), 365
\leftroot, 374
\leftskip, longitud, 201
\leftthreetimes (\leftthreetimes), 363
legalpaper, opción, 60
\leq (\leq), 364
leqno, opción, 359
\leqq (\leqq), 364
\leqslant (\leqslant), 364
\lessapprox (\lessapprox), 364
\lessdot (\lessdot), 363
\lesseqgtr (\lesseqgtr), 364
\lesseqqgtr (\lesseqqgtr), 364
\lessgtr (\lessgtr), 364
\lessim (\lessim), 364
\let, 467
letter, clase, 49
letterpaper, opción, 60
\lfloor (), 367
\lg (lg), 389
\lgem, 173
\lgem, 173
\lgroup (), 367
lignesbleues, opción, 454, 455
\lim (\lim), 389
\liminf (\liminf), 389
\limits, 393
\limsup (\limsup), 389
\linebreak, 161
\lineskip, longitud, 324
\lineskiplimit, longitud, 324
\linewidth, longitud, 225
linkbordercolor, opción, 406
linkcolor, opción, 406
linktocpage, opción, 411, 438

list, entorno, 281
\listfigurename, 171, 205
\listfiles, 211
\listoffigures, 86
\listoftables, 118
\listparindent, longitud, 284
\listtablename, 171, 205
\ll (<<), 364
\llap, 200, 318
\Lleftarrow (\Lleftarrow), 365
\lll (<<<), 364
\lmoustache (,), 367
\ln (ln), 389
\lnapprox (\lnapprox), 364
\lneq (\lneq), 364
\lneqq (\lneqq), 364
\lnsim (\lnsim), 364
LOF, fichero, 86, 205
\log (log), 389
LOG, fichero, 5
\Logo, 456
\long, 480
longitudes
 \abovecaptionskip, 303
 \abovedisplayshortskip,
 400
 \abovedisplayskip, 400
 \arraycolsep, 288
 \arrayrulewidth, 288
 \baselineskip, 162
 \belowcaptionskip, 303
 \belowdisplayshortskip,
 400
 \belowdisplayskip, 400
 \bigskipamount, 142
 \boxlength, 311
 \captionindent, 307
 \caption linewidth, 307
 \captionmargin, 306
 \captionwidth, 306
 \columnsep, 102
 \columnseprule, 102
 \columnwidth, 224
 \dashlength, 311
 \dblfloatsep, 303
 \dbltextfloatsep, 303
 \doublerulesep, 289
 \evensidemargin, 65, 224
 \extrarowheight, 291
 \floatsep, 303
 \footnotesep, 225

- \footruleskip, 220
 \footrulewidth, 220
 \footskip, 225
 \hangindent, 245, 473
 \headheight, 225
 \headrulewidth, 220
 \headsep, 225
 \headwidth, 218
 \hfuzz, 230
 \hoffset, 65, 224
 \hsize, 225
 \intextsep, 303
 \itemindent, 283
 \itemsep, 283
 \jot, 401
 \labelsep, 283
 \labelwidth, 283
 \leftmargin, 283
 \leftmargini, 283
 \leftmarginii, 283
 \leftmarginvi, 283
 \leftskip, 201
 \lineskip, 324
 \lineskiplimit, 324
 \linewidth, 225
 \listparindent, 284
 \marginparpush, 225
 \marginparsep, 225
 \marginparwidth, 225
 \mathindent, 401
 \medskipamount, 142
 \minimumskip, 436
 \minrowclearance, 301
 \multicolssep, 102
 \multilinegap, 384
 \oddsidemargin, 67, 224
 \overfullrule, 230
 \panelwidth, 442
 \paperheight, 225, 227
 \paperwidth, 225, 227
 \parfillskip, 161, 199, 201
 \parindent, 23
 \parsep, 284
 \parskip, 23
 \partopsep, 283
 \pdfpageheight, 225
 \pdfpagewidth, 226
 \premulticols, 102
 \rightmargin, 284
 \rightskip, 201
 \shadowthickness, 311
 \skip\footins, 226
 \smallskipamount, 142
 \tabcolsep, 288
 \textfloatsep, 303
 \textheight, 67, 224
 \textscreenwidth, 439
 \textwidth, 67, 224
 \topmargin, 226
 \topsep, 284, 400
 \topskip, 226
 \vfuzz, 230
 \voffset, 67, 224
 \Longleftarrow (\Leftarrow), 365
 \longleftarrow (\leftarrow), 365
 \Longleftrightarrow ($\Longleftarrow\rightarrow$), 365
 \longleftrightarrow ($\leftarrow\rightarrow$), 365
 \longmapsto (\rightarrow), 365
 \Longrightarrow (\Rightarrow), 365
 \longrightarrow (\rightarrow), 365
 longtable, entorno, 305
 longtable, paquete, 136, 292
 \loop, 483
 \looparrowleft ($\leftarrow\circlearrowleft$), 365
 \looparrowright ($\circlearrowright\rightarrow$), 365
 \looseness, 161
 LOT, fichero, 118, 205
 \lower, 321
 \lozenge (\lozenge), 362
 \lquoti, 176
 \lquotii, 176
 \lquotiii, 176
 lrbox, entorno, 153
 \lsc, 175
 \Lsh (\Lsh), 365
 \LTcapwidth, 293
 \ltimes (\ltimes), 363
 \LTpost, 293
 \LTpre, 293
 LTX, fichero, 4
 \lVert (\lVert), 367
 \lvert (\lvert), 367
 \lvertneqq ($\not\lvert\lvert$), 364
- M**
- \mainmatter, 56
 \makeatletter, 217
 \makeatother, 217
 \makebox, 147
 \makegloss, 271
 \makeglossary, 249
 makeidx, paquete, 233
 MAKEINDEX, ejecutable, 233, 240
 \makeindex, 233
 \makelabel, 285
 \MakeLowercase, 214
 \maketitle, 53
 \MakeUppercase, 214
 \mapsto (\rightarrow), 365
 \marginpar, 108
 \marginparpush, longitud, 225
 \marginparsep, longitud, 225
 \marginparwidth, longitud, 225
 \margins, 435
 \markboth, 212
 \markright, 212
 marvosym, paquete, 189
 math, entorno, 127, 358
 \mathbb (A), 362
 \mathbf (A), 362
 \mathbin, 361
 \mathcal (A), 362
 \mathchoice, 395
 \mathfrak (A), 362
 \mathindent, longitud, 401
 \mathit (A), 362
 \mathnormal (A), 362
 \mathop, 395
 \mathord, 361
 mathpazo, paquete, 185
 mathptmx, paquete, 134, 185, 363, 365, 393
 \mathrel, 361
 \mathring (A), 368
 \mathrm (A), 362
 \mathscr (A), 365
 mathscr, opción, 365
 \mathsf (A), 362
 \mathspanish, 175
 \mathtt (A), 362
 matrix, entorno, 379
 \max (máx), 389
 MaxMatrixCols, contador, 379
 \mbox, 147, 317
 \mdseries (A), 43
 \measuredangle (\angle), 362
 \medskip, 68
 \medskipamount, longitud, 142
 \medspace, 134, 360
 menubordercolor, opción, 406
 menucolor, opción, 406
 MF, fichero, 178
 \mho (U), 361
 \mid (\mid), 366

- MiK_TE_X, 11
\min (mín), 389
\minimumskip, longitud, 436
minipage, entorno, 149
minitoc, paquete, 272
\minrowclearance, longitud, 301
mm, unidad de longitud, 143
\mod, 390
\models (|=), 366
monochrome, opción, 273
\moveleft, 321
\moveright, 321
\mp (±), 363
mpfootnote, contador, 150
mpmulti, paquete, 445
mtcoff, paquete, 272
\mu (μ), 134
mu, unidad de longitud, 143
multicol, paquete, 102, 136
multicolpar, entorno, 103
multicolpar, paquete, 102, 103
multicols, entorno, 102
\multicolsep, longitud, 102
\multicolumn, 116
\multido, 349
multido, paquete, 349
\multimap (→), 365
\multiply, 470
\multips, 349
\multirow, 295
multirow, paquete, 295
\multirowsetup, 296
\multirput, 348
multiline, entorno, 384
\multilinegap, longitud, 384
\mvchr, 192
- N**
- \nabla (∇), 362
namelimits, opción, 359
\naput, 351
natheight, opción, 337
\natural (\natural), 362
natwidth, opción, 337
navabar, opción, 440
\navabarBgColor, 441
\NaviBarOff, 440
\NaviBarOn, 440
\navabarTextColor, 441
\nbput, 351
\ncangle, 351
\ncangles, 351
\ncarc, 351
\ncarcbox, 351
\ncbar, 351
\ncbox, 351
\nccircle, 351
\nccurve, 351
\ncdiag, 351
\ncdiagg, 351
\ncline, 351
\ncloop, 351
\ncong (⊐), 364
\ncput, 351
\nearrow (↗), 365
\neg (¬), 362
\negmedspace, 134
\negthickspace, 134
\negthinspace, 68, 134
\neq (≠), 364
\newblock, 253
\newcaptionstyle, 307
newcent, paquete, 187
\newcolumntype, 290
\newcommand, 95
\newcommand*, 94
\newcount, 468
\newcounter, 195
\newdimen, 468
\newenvironment, 95
\newenvironment*, 95
\newif, 482
\newindex, 246
\newlength, 196
\newline, 23
\newmuskip, 468
\newNaviIcon, 443
\NewPage, 444
\newpage, 69, 102, 163, 164
\newsavebox, 152
\newskip, 468
\newtheorem, 397
\newtheorem*, 398
\nexists (\exists), 362
\ngeq (\geq), 364
\ngeqq (\geqslant), 364
\ngeqlant (\geqslant), 364
\ngr (\triangleright), 364
\ni (\ni), 366
\Leftarrow (\Leftarrow), 365
\leftarrow (\leftarrow), 365
\Leftarrowrightarrow ($\Leftarrow\Rightarrow$), 365
\Leftrightarrowrightarrow ($\Leftarrow\Rightarrow$), 365
\nleq (\leq), 364
\nleqq (\leqslant), 364
\nleqlant (\leqslant), 364
\nless (\lessdot), 364
\nmid (\mid), 366
noaccumulate, opción, 455
\nobreakdash, 370
\nocite, 252
nocolorBG, opción, 455
nodirectory, opción, 437
\noindent, 23
\nointerlineskip, 324
nointlimits, opción, 359
\nolimits, 393
\nolinebreak, 161
nomarkers, opción, 446
nonamelimits, opción, 359
\nonfrenchspacing, 160
\nonumber, 381
\nopagebreak, 163
\nopicchangepage, 310
\normalmarginpar, 109
\normalsize, 44
norsk, opción, 434
nosumlimits, opción, 359
\not, 360
\notag, 381, 382
\notin (\notin), 366
notitlepage, opción, 61
nototal, opción, 455
\nouppercase, 221
\nparallel (\parallel), 366
\nprec (\prec), 364
\npreceq (\preceq), 364
\nrightarrow (\rightarrow), 365
\nrightarrowarrow (\rightarrowtail), 365
\nshortmid (\shortmid), 366
\nshortparallel (\shortparallel), 366
\nsim (\sim), 364
\nsubseteq (\subseteq), 366
\nsubseteqq (\subseteq), 366
\nsucc (\succ), 364
\nsucceq (\succ), 364
\nsupseteq (\supseteq), 366
\nsupseteqq (\supseteq), 366
\ntriangleleft (\triangleleft), 366
\ntrianglelefteq (\trianglelefteq), 366
\ntriangleright (\triangleright), 366
\ntrianglerighteq (\trianglerighteq), 366
\nu (v), 134
nuancegris, opción, 454, 455

\null, 198
\numberline, 207
\numberwithin, 388
\nVDash (¥), 366
\nVdash (¥), 366
\nvDash (¥), 366
\nvDash (¥), 366
\nwarrow (↖), 365

O
\obeylines, 202
\obeyspaces, 202
\OBox, 447

\oddsidemargin, longitud, 67, 224
\odot (○), 363

\OE, 38
\oe, 38

\offinterlineskip, 324
\oint (∮), 393

\oldstylenums, 130
\Omega (Ω), 134

\omega (ω), 134
\ominus (⊖), 363

\onecolumn, 101
onecolumn, opción, 61

\onelinecaption, 309
oneside, opción, 61

\OnlySlide, 460
\onlySlide, 460

\onlySlide*, 460
opciones

10pt, 59
11pt, 59
12pt, 59

a4paper, 60
a5paper, 60

accumulate, 455
activeacute, 171

activegrave, 171
alien, 454

alienglow, 455
anchorcolor, 406

angle, 337
autumn, 454, 455

azure, 454, 455
b5paper, 60

backref, 411
baseurl, 411

basque, 166, 171
bb, 336

bgadd, 448
bookmarks, 407

bookmarksnumbered, 407
bookmarksopen, 407
bookmarksopenlevel, 407
breaklinks, 405
catalan, 165, 166, 171
centertags, 358
citebordercolor, 406
citecolor, 406
colorBG, 455
colorlinks, 75, 406
command, 337
contemporain, 454, 455
dansk, 434
darkblue, 454, 455
designi, 435
designii, 435
designiii, 435
designiv, 435
designv, 435
draft, 60, 75, 335, 455, 508, 513
dutch, 434
dvipdf, 274
dvipdfm, 274, 434, 445
dvips, 45, 74, 274, 434
dvipsnames, 46
dvipsone, 274, 434
dviwin, 274
dviwindo, 274, 434
emtex, 274
executivepaper, 60
ext, 337
extension, 411
filebordercolor, 406
filecolor, 406
final, 60, 455
final|draft, 335
finnish, 434
fleqn, 359, 401
forcolorpaper, 444
forpaper, 444
frames, 454, 455
french, 434
frenchlinks, 405
galician, 166, 171
german, 434
gyom, 454, 455
hiderotate, 336
hidescale, 336
hiresbb, 336, 337
hyperfigures, 411
hyperindex, 411

ignore, 445
intlimits, 359
italian, 434
landscape, 227
latextoc, 437
latin1, 16, 34
leftpanel, 441
legalpaper, 60
leqno, 359
letterpaper, 60
lignesbleues, 454, 455
linkbordercolor, 406
linkcolor, 406
linktocpage, 411, 438
mathscr, 365
menubordercolor, 406
menucolor, 406
monochrome, 273
namelimits, 359
natheight, 337
natwidth, 337
navibar, 440
noaccumulate, 455
nocolorBG, 455
nodirectory, 437
nointlimits, 359
nomarkers, 446
nonamelimits, 359
norsk, 434
nosumlimits, 359
notitlepage, 61
nototal, 455
nuancegris, 454, 455
onecolumn, 61
oneside, 61
openany, 61
openbib, 255
openright, 61
origin, 337
OT1, 32
oztex, 274
pageanchor, 405
pagebackref, 411
pagebordercolor, 406
pagecolor, 406
pascal, 454, 455
pctex32, 274
pctexhp, 274
pctexps, 274
pctexwin, 274
pdf, 455

- pdfauthor, 409
 pdfborder, 406
 pdfcenterwindow, 408
 pdfcreator, 409
 pdffitwindow, 408
 pdfhighlight, 409
 pdfkeywords, 409
 pdfmenubar, 408
 pdfpagelayout, 408
 pdfpagemode, 407
 pdfpagescrop, 409
 pdfpagetransition, 408
 pdfproducer, 409
 pdfstartpage, 408
 pdfstartview, 408
 pdfsubject, 409
 pdftex, 45, 74, 274, 434
 pdftitle, 409
 pdftoolbar, 408
 pdfwindowui, 408
 plainpages, 406
 polish, 434
 ps, 455
 raiselinks, 405
 read, 337
 reqno, 359
 rico, 454, 455
 rightpanel, 441, 442
 russian, 434
 scaled, 184
 slantedGreek, 185
 slideBW, 455
 slideColor, 455
 spanish, 17, 130, 165, 166, 171,
 391, 434
 sumlimits, 359
 T1, 17, 32
 tbtags, 358
 tcidvi, 274
 textures, 274, 434
 tight, 444
 titlepage, 61
 total, 455
 totalheight, 336
 trim, 337
 troispoints, 454, 455
 truetex, 274
 twocolumn, 61, 101, 481
 twoside, 61, 481
 type, 337
 urlbordercolor, 406
 urlicolor, 406
 usenames, 46
 usetemplates, 438, 441
 viewport, 337
 xdiv, 274
 openany, opción, 61
 openbib, opción, 255
 openright, opción, 61
 \oplus (\oplus), 363
 \optionalpagematter, 436
 origin, opción, 337
 \oslash (\oslash), 363
 OT1, opción, 32
 otherlanguage, entorno, 167
 otherlanguage*, entorno, 167
 \otimes (\otimes), 363
 \outer, 480
 \ovalbox, 148
 \overbrace, 372
 \overfullrule, longitud, 230
 \overlays, 459
 \overleftarrow, 373
 \overrightarrow, 373
 \overline, 372
 \overrightarrow, 373
 \overset, 373
 oztex, opción, 274
 OzTeX, 11
- P**
- \P (\P), 38
 page, contador, 139
 pageanchor, opción, 405
 pagebackref, opción, 411
 pagebordercolor, opción, 406
 \pagebreak, 163
 \pagecolor, 275, 448
 pagecolor, opción, 406
 pageframe, paquete, 226
 \pagename, 171
 \pageref, 72, 128
 \pagestyle, 64
 \pageTransitionGlitter, 447
 palatino, paquete, 184, 187
 \panelBgColor, 439
 \panelNaviGroup, 442
 \panelsep, 442
 \paneltemplate, 439
 \panelwidth, longitud, 442
 \paperheight, longitud, 225, 227
 \paperwidth, longitud, 225, 227
 paquetes
 accents, 369
 achicago, 266
 afterpag, 304
 amsbsy, 357
 amscd, 395
 amsfonts, 357
 amsmath, 126, 357
 amsopn, 357
 amsref, 402
 amssymb, 357, 434, 452
 amstex, 136, 177
 amstext, 357
 amsthm, 398
 array, 289, 378
 avant, 187
 babel, 17, 130, 164, 391, 435
 background, 445, 447
 bibunits, 268
 bookman, 187
 calc, 487
 calrsfs, 365
 caption2, 305
 chancery, 187
 chemsym, 135
 cite, 266
 citesort, 266
 cm-super, 17
 color, 45, 273, 434
 colortbl, 298
 comprehensive, 192
 courier, 187
 dcolumn, 120, 328
 delarray, 379
 diagram, 395
 dijslib, 461
 doublespace, 29
 eepic, 356
 endnotes, 329
 epic, 356
 epstopdf, 335
 eso-pic, 434, 448
 eucal, 365
 eurosym, 39, 192
 everyshi, 434
 exerquiz, 461
 extramarks, 223
 fancybox, 148, 326, 372
 fancyhdr, 137, 217
 float, 328
 fontenc, 17, 180
 footbib, 266

- fp, 488
gloss, 271
graphicx, 80, 273, 335, 355, 452
helvet, 184
hhline, 296
hyperref, 74, 273, 434, 452
ifthen, 177
index, 246
inputenc, 16, 202
insdls, 461
kuvio, 395
lastpage, 222
layout, 226
longtable, 136, 292
makeidx, 233
marvosym, 189
mathpazo, 185
mathptmx, 134, 185, 363, 365, 393
minitoc, 272
mpmulti, 445
mtcoff, 272
multicol, 102, 136
multicolpar, 102, 103
multido, 349
multirow, 295
newcent, 187
pageframe, 226
palatino, 184, 187
pause, 446
pdfscreen, 434
pdftricks, 335
picinpar, 329
picins, 310
pifont, 189
pp4link, 449
problema, 264
ps4pdf, 335
PSfrag, 334
pst-3d, 355
pst-3dplot, 355
pst-all, 355
pst-char, 355
pst-circ, 355
pst-coil, 355
pst-eps, 355
pst-fill, 355
pst-fr3d, 355
pst-gr3d, 355
pst-grad, 355
pst-lens, 355
pst-node, 355
pst-osci, 355
pst-plot, 355
pst-poly, 355
pst-text, 355
pst-tree, 355
pst-vue3D, 355
pstcol, 355
PSTricks, 452, 461
pstricks, 355
rotating, 136, 340, 355
seminar, 452
shapepar, 29
showidx, 248
subfigure, 329
tabularx, 328
tex4ht, 418
times, 184, 187, 452
titleref, 272
varindex, 239
varioref, 272
vmargin, 226
web, 434
wrapfig, 314
xr, 272
xypic, 395
\par, 23
\paragraph, 50
\paragraph*, 50
\parallel (||), 366
\parbox, 148, 320
\parfillskip, longitud, 161, 199, 201
\parindent, longitud, 23
\parpic, 310
\parrafo, contador, 475
\parsep, longitud, 284
\parshape, 474
\parskip, 144
\parskip, longitud, 23
\part, 458
\part, contador, 205
\part, 50
\part*, 50
\partial (∂), 361
\partname, 171, 205
\partopsep, longitud, 283
pascal, opción, 454, 455
\pause, 446, 449
pause, paquete, 446
\pausebuild, 452
\pausecolorreset, 451
\pausecolors, 451
\pauseDissolve, 447
\pauseGlitter, 447
\pauseHBlinds, 447
\pausehighlight, 452
\pauseHSplit, 447
\pauseHOSplit, 447
\pauseIBox, 447
\pausellevel, 450
\pauseOBox, 447
\pauseReplace, 447
\pauseVBlinds, 447
\pauseVISplit, 447
\pauseVOSplit, 447
\pauseWipe, 447
pc, unidad de longitud, 143
pctex32, opción, 274
pctexhp, opción, 274
pctexps, opción, 274
pctexwin, opción, 274
pdf, opción, 455
PDF, fichero, 7, 8
\pdfannot, 413
pdfauthor, opción, 409
pdfborder, opción, 406
pdfcenterwindow, opción, 408
pdfcreator, opción, 409
pdffitwindow, opción, 408
pdfhighlight, opción, 409
pdfkeywords, opción, 409
\pdflastannot, 413
PDFLATEX, compilador, 8, 15
pdfmenubar, opción, 408
\pdfpageheight, longitud, 225
pdfpagelayout, opción, 408
pdfpagemode, opción, 407
pdfpagescrop, opción, 409
pdfpagetransition, opción, 408
\pdfpagewidth, longitud, 226
pdfproducer, opción, 409
pdfscreen, paquete, 434
pdfstartpage, opción, 408
pdfstartview, opción, 408
\pdfstringdef, 409
pdfsubject, opción, 409
pdftex, opción, 45, 74, 274, 434
pdftitle, opción, 409
pdftoolbar, opción, 408
pdftricks, paquete, 335
pdfwindowui, opción, 408

- \@enumi, 278
 \@enumii, 278
 \@enumiii, 278
 \@enumiv, 278
 \perp, 366
 PFB, fichero, 182
 \phantom, 372
 \Phi (Φ), 134
 \phi (ϕ), 134
 \Pi (Π), 134
 \pi (π), 134
 Piautolist, entorno, 191
 \piccaption, 312
 \piccaptioninside, 313
 \piccaptionoutside, 313
 \piccaptionside, 313
 \piccaptiontopside, 313
 \picchangemode, 310
 \pichskip, 311
 picinpar, paquete, 329
 picins, paquete, 310
 \picskip, 312
 picture, entorno, 148, 355
 \Picture+, 427
 \Pifill, 191
 pifont, paquete, 189
 \Piline, 191
 Pilist, entorno, 191
 \Pisymbol, 191
 \pitchfork (\pitchfork), 366
 PK, fichero, 178
 PL, fichero, 178
 plain, estilo de página, 211
 plainpages, opción, 406
 PLTOTF, ejecutable, 178
 \pm (\pm), 363
 pmatrix, entorno, 379
 \pmb, 366, 367
 \pmod, 390
 \pod, 390
 poema, entorno, 324
 polish, opción, 434
 pp4link, paquete, 449
 \ppleuro, 186
 \Pr (\Pr), 389
 \Preamble, 424
 \prec (\prec), 364
 \preccapprox (\preccapprox), 364
 \preccurlyeq (\preccurlyeq), 364
 \preceq (\preceq), 364
 \precnapprox (\precnapprox), 364
 \precneqq (\precneqq), 364
 \precnsim (\precnsim), 364
 \precsim (\precsim), 364
 \premulticols, longitud, 102
 \prime (), 362
 \printgloss, 271
 \printindex, 233
 problema, paquete, 264
 proc, clase, 49
 \prod (\prod), 393
 \projlim (projlim), 389
 proof, entorno, 399
 \proofmodefalse, 248
 \proofmodetru, 248
 \proofname, 171, 400
 \proto (\propto), 366
 prosper, clase, 452
 \protect, 207, 208
 \providecommand, 95
 \providecommand*, 94
 ps, opción, 455
 PS, fichero, 6
 PS2PDF, ejecutable, 334
 ps4pdf, paquete, 335
 \psarc, 346
 \psbezier, 343
 \psccurve, 344
 \pscircle, 346
 \pscirclebox, 348
 \pscurve, 344
 \psdblframebox, 348
 \psdiabox, 348
 \psdots, 343
 \psellipse, 344
 \psfrag, paquete, 334
 \psframe, 345
 \psframebox, 348
 \psgrid, 346
 \Psi (Ψ), 134
 \psi (ψ), 134
 \psline, 343
 \psmatrix, entorno, 352
 \psovalbox, 348
 \pspicture, entorno, 341
 \pspolygon, 345
 \psset, 341
 \psshadowbox, 348
 pst-3d, paquete, 355
 pst-3dplot, paquete, 355
 pst-all, paquete, 355
 pst-char, paquete, 355
 pst-circ, paquete, 355
 pst-coil, paquete, 355
 pst-eps, paquete, 355
 pst-fill, paquete, 355
 pst-fr3d, paquete, 355
 pst-gr3d, paquete, 355
 pst-grad, paquete, 355
 pst-lens, paquete, 355
 pst-node, paquete, 355
 pst-osci, paquete, 355
 pst-plot, paquete, 355
 pst-poly, paquete, 355
 pst-text, paquete, 355
 pst-tree, paquete, 355
 pst-vue3D, paquete, 355
 PST2PDF, ejecutable, 335
 pstcol, paquete, 355
 \pstbox, 348
 PSTricks, paquete, 452, 461
 psticks, paquete, 355
 \pswedge, 346
 pt, unidad de longitud, 143
 \putbib, 269
- Q**
- \qed, 400
 \qedhere, 400
 \quad, 68
 \quad, 68
 quotation, entorno, 27
 quote, entorno, 27
 quoting, entorno, 175
- R**
- \radians, 342
 \raggedbottom, 164
 \raggedleft, 27
 \raggedright, 27
 \raise, 321
 \raisebox, 151
 raiselinks, opción, 405
 \raisetag, 383
 \rangle (), 367
 \rbrace (), 367
 \rceil (), 367
 \Re (\Re), 361
 read, opción, 337
 \readdata, 344
 \ref, 72, 128, 196
 \reflectbox, 339
 \refname, 171, 250

- \refstepcounter, 196
 \renewcaptionstyle, 307
 \renewcommand, 95
 \renewcommand*, 94
 \renewenvironment, 95
 \renewenvironment*, 95
 \renewindex, 246
 \repeat, 483
 \Replace, 447
 report, clase, 49
 reqno, opción, 359
 \resizebox(*), 339
 \reversemarginpar, 109
 \rfloor, 367
 \rgroup (J), 367
 \rho (ρ), 134
 rico, opción, 454, 455
 \right, 376
 \right., 376
 \Rightarrow (\Rightarrow), 365
 \rightarrowarrow (\rightarrow), 365
 \rightarrowfill, 199
 \rightarrowtail (\rightarrowtail), 365
 \rightharpoondown (\rightarrowtail), 365
 \rightharpoonup (\rightarrowtail), 365
 \rightleftarrows (\rightleftarrows), 365
 \rightleftharpoons (\rightleftharpoons), 365
 \rightline, 26, 318
 \rightmargin, longitud, 284
 \rightmark, 212
 rightpanel, opción, 441, 442
 \rightrightarrows (\rightleftarrows), 365
 \rightskip, longitud, 201
 \rightsquigarrow (\rightsquigarrow), 365
 \rightthreetimes (\rightthreetimes), 363
 \risingdotseq (\eqqcolon), 364
 \rlap, 200, 318
 \rmfamily (A), 42
 \rmoustache (,), 367
 \Roman, 140
 \roman, 140, 174
 \rotatebox, 338
 rotating, paquete, 136, 340, 355
 \rotcaption, 340
 \rowcolor, 299
 \rput, 347
 \rquoti, 176
 \rquotii, 176
 \rquotiii, 176
 \Rrightarrow (\Rrightarrow), 365
 \Rsh (F), 365
- \rtimes (\rtimes), 363
 \rule, 150
 russian, opción, 434
 \rVert (||), 367
 \rvert (|), 367
- S**
- \S (\$), 38
 \samepage, 163
 \savebox, 152
 \savedata, 344
 \sbox, 152
 \scalebox, 339
 scaled, opción, 184
 \screensize, 435
 \scriptscriptstyle, 371
 \scriptsize, 44
 \scriptstyle, 371
 \scshape (A), 42
 \searrow (\searrow), 365
 \sec (sec), 389
 \secnumdepth, contador, 204
 section, contador, 139, 204
 \section, 50
 \section*, 50
 \sectionmark, 51
 \see, 235
 \seealso, 235
 \seename, 171, 236
 \selectfont, 194
 \selectlanguage, 167
 \selectspanish, 174
 \selectspanish*, 176
 seminar, paquete, 452
 \sen (sen), 390
 \senh (senh), 391
 \setbox, 323
 \setcaptionmargin, 305
 \setcaptionwidth, 305
 \setcounter, 141, 466
 \setlength, 143, 466
 \setminus (\setminus), 363
 \settodepth, 197
 \settoheight, 197
 \settowidth, 197
 \sffamily (A), 42
 \shadowbox, 148
 \shadowsize, 148
 \shadowthickness, longitud, 311
 shapepar, paquete, 29
 \sharp (#), 362
 \shorthandoff, 170
- \shorthandon, 170
 \shorthandsspanish, 175
 \shortindexingoff, 248
 \shortindexingon, 248
 shortindexingon, entorno, 248
 \shortmid (|), 366
 \shortparallel (||), 366
 \shoveleft, 384
 \shoveright, 384
 \showcols, 291
 showidx, paquete, 248
 \sideset, 373
 sidewaysfigure, entorno, 340
 sidewaystable, entorno, 340
 \Sigma (Σ), 134
 \sigma (σ), 134
 \sim (~), 364
 \simeq (\simeq), 364
 \sin (sin), 389
 \sinh (sinh), 389
 skip, 196
 \skip\footins, longitud, 226
 slantedGreek, opción, 185
 slide, entorno, 453, 457
 slideBW, opción, 455
 \slideCaption, 456
 slideColor, opción, 455
 slides, clase, 49
 \sloppy, 230
 sloppypar, entorno, 230
 \slshape (A), 42
 \small, 44
 \smallfrown (~), 366
 \smallint (f), 393
 smallmatrix, entorno, 380
 \smallsetminus (\setminus), 363
 \smallskip, 68
 \smallskipamount, longitud, 142
 \smallsmile (~), 366
 \smash, 374
 \smile (~), 366
 sp, unidad de longitud, 143
 \spacedoperators, 391
 \spadesuit (♠), 362
 spanish, opción, 17, 130, 165, 166,
 171, 391, 434
 \spanishdeactivate, 176
 \spanishdecimal, 392
 \spanishoperators, 391, 392
 \special, 228, 331
 \SpecialCoor, 342

- \sphericalangle (\sphericalangle), 362
 \split, entorno, 358, 383
 \sptext, 37, 172, 175
 \sqcap (\sqcap), 363
 \sqcup (\sqcup), 363
 \sqrt (\sqrt), 131, 374
 \sqsubset (\sqsubset), 366
 \sqsubseteqq (\sqsubseteq), 366
 \sqsupset (\sqsupset), 366
 \sqsupseteqq (\sqsupseteq), 366
 \square (\square), 362
 \stackrel, 373
 \star (*), 363
 \stepcounter, 196
 \stretch, 198
 subarray, entorno, 394
 subequations, entorno, 388
 subfigure, paquete, 329
 \subitem, 233
 \subject, 436
 \ subparagraph, 50
 \ subparagraph*, 50
 subsection, contador, 205
 \subsection, 50
 \subsection*, 50
 \subsectionmark, 51
 \Subset (\Subset), 366
 \subset (\subset), 366
 \subsetneqq (\subsetneqq), 366
 \subsetneqqq (\subsetneqqq), 366
 \substack, 393
 \subsubitem, 233
 subsubsection, contador, 205
 \subsubsection, 50
 \subsubsection*, 50
 \subtitle, 456
 \succ (>), 364
 \succapprox (\succapprox), 364
 \succcurlyeq (\succcurlyeq), 364
 \succeq (\succeq), 364
 \succnapprox (\succnapprox), 364
 \succneqq (\succneqq), 364
 \succnsim (\succnsim), 364
 \succsim (\succsim), 364
 \sum (\sum), 393
 sumlimits, opción, 359
 \sup (sup), 389
 \suppressfloats, 304
 \Supset (\Supset), 366
- \upset (\upset), 366
 \upseteqq (\upseteqq), 366
 \upsetneqq (\upsetneqq), 366
 \upsetneqqq (\upsetneqqq), 366
 \surd (\surd), 362
 \swapnumbers, 399
 \swarrow (\swarrow), 365
 \symbol, 189
 symbols-a4.pdf, 192
- T**
- T1, opción, 17, 32
 T4HT, ejecutable, 418
 \tabcolsep, longitud, 288
 table, contador, 118
 table*, entorno, 119
 \tablename, 171
 \tableofcontents, 57, 428, 437
 tabular, entorno, 111
 \tabularnewline, 111
 tabularx, paquete, 328
 \tag, 382
 \tag*, 382
 \tan (tan), 389
 \tanh (tanh), 389
 \tau (τ), 134
 \tbinom, 133, 375
 tbtags, opción, 358
 tcidvi, opción, 274
 \template, 438
 teTeX, 11
 \TeX, 13
 TEX, fichero, 4
 TEX4HT, ejecutable, 418
 tex4ht, paquete, 418
 \texorpdfstring, 410, 436
 TeXShop, 11
 \text, 133, 387
 \textasciicircum, 32
 \textasciitilde, 32
 \textbf (A), 43
 \textBgColor, 439
 \textbullet (•), 38
 \textcircled (Ⓐ), 38
 \textcolor, 46, 275
 \textdegree (°), 38
 \textfloatsep, longitud, 303
 \textfraction, 302
 \textheight, longitud, 67, 224
 \textit (A), 42
 \textmd (A), 43
- \TexttoEPS, entorno, 350
 \textregistered (®), 38
 \textrm (A), 42
 \textsc (A), 42
 \textscreenwidth, longitud, 439
 \textsf (A), 42
 \textsl (A), 42
 \textspanish, 175
 \textstyle, 132, 371
 \textsuperscript, 37
 \texttrademark (™), 38
 \texttt (A), 42
 \textup (A), 42
 textures, opción, 274, 434
 \Textures, 11
 \textvisiblespace („), 38
 \textwidth, longitud, 67, 224
 TFM, fichero, 178
 \tfrac, 132
 TFTOPL, ejecutable, 178
 \tg (tg), 390
 \tgh (tgh), 391
 \thanks, 53
 \the, 140, 142, 195
 \bibliography, entorno, 123
 \gloss, entorno, 271
 \glossary, entorno, 249
 \index, entorno, 233
 \theoremstyle, 398
 \therefore (∴), 366
 \Theta (Θ), 134
 \theta (θ), 134
 \thickapprox (\thickapprox), 364
 \thicklines, 148
 \thicksim (~), 364
 \thickspace, 134, 360
 \thinlines, 148
 \thinspace, 68, 134, 360
 \thisfancyput, 326
 \thisfancyput*, 327
 \thispagestyle, 64
 tight, opción, 444
 \tilde (˜), 368
 \time, 470
 \times (×), 363
 times, paquete, 184, 187, 452
 \tiny, 44
 \title, 53, 436, 456
 \titleauthorproportion, 436
 titlepage, entorno, 54

- titlepage, opción, 61
 titeref, paquete, 272
 TKBIBTEX, ejecutable, 265
 TOC, fichero, 57, 205
 \TocAt, 428
 tocdepth, contador, 204
 \today, 15, 478
 \top (⊤), 362
 \topfigrule, 304
 \topfraction, 302
 \toplink, 449
 \topmargin, longitud, 226
 topnumber, contador, 302
 \topsep, longitud, 284, 400
 \topskip, longitud, 226
 \toptarget, 449
 total, opción, 455
 totalheight, opción, 336
 totalnumber, contador, 302
 \triangle (△), 362
 \triangledown (▽), 362
 \triangleleft (◁), 363
 \trianglelefteq (⊲), 366
 \triangleq (△≡), 364
 \triangleright (▷), 363
 \trianglerighteq (▷=), 366
 trim, opción, 337
 trivlist, entorno, 284
 troispoints, opción, 454, 455
 trueTeX, opción, 274
 \ttfamily (A), 42
 \twocolumn, 101
 twocolumn, opción, 61, 101, 481
 \twoheadleftarrow (↔), 365
 \twoheadrightarrow (↔), 365
 twoside, opción, 61, 481
 type, opción, 337
- U**
 \unaccentedoperators, 391
 \underbrace, 372
 \underleftarrow, 373
 \underleftrightarrow, 373
 \underline, 44, 372
 \underrightarrow, 373
 \underset, 373
 \unhbox, 323
 \unhcopy, 323
 \university, 436
 \unskip, 202
 \unspacedoperators, 391
 \UntilSlide, 460
- \untilSlide, 460
 \untilSlide*, 460
 \unvbox, 323
 \unvcopy, 323
 \up, 173
 \Uparrow (↑), 365
 \uparrow (↑), 365
 \upbracefill, 199
 \Updownarrow (↕), 365
 \updownarrow (↓), 365
 \upharpoonleft (↼), 365
 \upharpoonright (↼), 365
 \uplus (⊕), 363
 \uproot, 374
 \upshape (A), 42
 \Upsilon (Υ), 134
 \upsilon (υ), 134
 \upuparrows (↑↑), 365
 \uput, 347
 urlbordercolor, opción, 406
 urlcolor, opción, 406
 \usebox, 152
 \usecaptionmargin, 307
 \usecounter, 285
 usenames, opción, 46
 \usepackage, 16
 \useshorthands, 170
 \usetemplates, opción, 438, 441
- V**
- \value, 195
 \varDelta (Δ), 361
 \varEpsilon (ε), 134
 \varGamma (Γ), 361
 varindex, paquete, 239
 \varinjlim (lim), 389
 varioref, paquete, 272
 \varLambda (Λ), 361
 \varliminf (lim), 389
 \varlimsup (lim), 389
 \varnothing (Ø), 362
 \varOmega (Ω), 361
 \varPhi (Φ), 361
 \varphi (φ), 134
 \varPi (Π), 361
 \varpi (ϖ), 134
 \varprojlim (lim), 389
 \varpropto (∞), 366
 \varPsi (Ψ), 361
 \varrho (ϱ), 134
 \varSigma (Σ), 361
 \varsigma (ς), 134
- \varsubsetneq (⊈), 366
 \varsubsetneqq (⊉), 366
 \varsupsetneq (⊇), 366
 \varsupsetneqq (⊈), 366
 \varTheta (Θ), 361
 \vartheta (ϑ), 134
 \vartriangle (△), 366
 \vartriangleleft (◁), 366
 \vartriangleright (▷), 366
 \varUpsilon (Υ), 361
 \varXi (Ξ), 361
 \VBlinds, 447
 \vbox, 319
 \vcenter, 320
 \Vdash (Vdash), 366
 \vDash (Vdash), 366
 \vdash (Vdash), 366
- \vdots (⋮), 370
 \vec (x), 368
 \vee (∨), 363
 \veebar (⊎), 363
 \verb, 202
 verbatim, entorno, 202
 verse, entorno, 28
 \version, 436
 \Vert (||), 367
 \vert (|), 367
 \vfil, 199
 \vfill, 153, 199
 \vfilneg, 199
 \vfuzz, longitud, 230
 viewport, opción, 337
 \VISplit, 447
 vmargin, paquete, 226
 Vmatrix, entorno, 379
 vmatrix, entorno, 379
 \voffset, 64
 \voffset, longitud, 67, 224
 \VOSplit, 447
 \vpagecolor, 448
 \vphantom, 372
 \vrule, 322
 \vrulefill, 472
 \vspace, 68
 \vspace*, 68
 \vss, 199
 \vtop, 320
 \VvDash (VvDash), 366
- W**
- \wd, 323

- web, paquete, 434
\web@article, 435
\webauthor, 437
\web@back, 435
\web@continued, 435
\web@copyright, 435
\web@directory, 435
\web@doc, 435
\webemail, 437
\webkeywords, 437
\web@revision, 435
\web@section, 435
\websubject, 437
\webtitle, 437
\web@toc, 435
\webuniversity, 437
\webversion, 437
- \web@versionlabel, 435
\wedge (\wedge), 363
\@whiledim, 483
\@whilenum, 483
\@whilesw, 484
\widehat (\widehat{x}), 368
\widetilde (\widetilde{x}), 368
\widowpenalty, 67
WINBIBDB, ejecutable, 265
\Wipe, 447
\wp (\wp), 361
\wr (\wr), 363
wrapfig, paquete, 314
- X**
- \xdef, 467
XDVI, ejecutable, 179
- xdvi, opción, 274
\Xi (Ξ), 134
\xi (ξ), 134
XINDY, ejecutable, 238
\xleaders, 472
\xleftarrow, 373
\xpar, 103
xr, paquete, 272
\xrightarrow, 373
xypic, paquete, 395
- Y**
- YAP, ejecutable, 81, 179
- Z**
- \zeta (ζ), 134

Bibliografía

- [1] ARSENEAU, D.: «The cite package», documento distribuido con el paquete, 1997.
- [2] BEZOS, J.: «The accents package», documento distribuido con el paquete, 1999.
- [3] _____ «Estilo spanish para el paquete babel (spanish package)», documento distribuido con el paquete, 2000.
- [4] BLESER, J.: «The picins package», documento distribuido con el paquete, 1992.
- [5] BRAAMS, J., CARLISLE, D., JEFFREY, A., LAMPORT, L., MITTELBACH, F., ROWLEY, C. y SCHÖPF, R.: «Standard (L^AT_EX) packages makeidx and showidx», documento distribuido con el paquete, 1994.
- [6] BRAAMS, J.: «Babel, a multilingual style-option system for use with L^AT_EX's standard document styles», *TUGboat* **12** (1991), 291–301.
- [7] CARLISLE, D.: «The hhline package», documento distribuido con el paquete, 1994.
- [8] _____ «The afterpage package», documento distribuido con el paquete, 1995.
- [9] _____ «The longtable package», documento distribuido con el paquete, 1995.
- [10] _____ «Packages in the “graphics” bundle», documento incluido en la distribución de L^AT_EX, 1998.
- [11] CASCALES SALINAS, B., LUCAS SAORÍN, P., MIRA ROS, J. M., PALLARÉS RUÍZ, A. y SÁNCHEZ-PEDREÑO GUILLÉN, S.: *L^AT_EX, una imprenta en sus manos*, ADI, Madrid, 2000.
- [12] CHAMBERT-LOIR, A., CV, R. y CV, R.: «The pdftricks package», documento distribuido con el paquete, 2001.
- [13] CHENG, P., HARRISON, M. y RODGERS, R.: «Makeindex — A general purpose, formatter-independent index processor», documento distribuido con el programa, 1991.
- [14] *The Chicago Manual of Style*, University of Chicago Press, 1982.
- [15] DAHLGREN, M.: «Chemsym — A L^AT_EX macro for chemical symbols», documento distribuido con el paquete, 1998.
- [16] DÍAZ, J. L. y BEZOS, J.: «The gloss package», documento distribuido con el paquete, 2002.
- [17] DOMENJOU, E.: «The footbib package», documento distribuido con el paquete, 1999.
- [18] DRAKOS, N.: «The (L^AT_EX)2html translator», documento distribuido con el programa, 1999.
- [19] FERNÁNDEZ, J. A.: «The bibunits package», documento distribuido con el paquete, 1991.
- [20] GIROU, D. y RAHTZ, S.: «PSTricks 97», documento distribuido con el paquete, 1998.
- [21] GOOSSENS, M., MITTELBACH, F. y SAMARIN, A.: *The L^AT_EX Companion*, Addison-Wesley, 1994.
- [22] GOOSSENS, M., RAHTZ, S., GURARI, E., MOORE, R. y SUTOR, R.: *The L^AT_EX Web Companion*, Addison-Wesley, 1986.
- [23] GOOSSENS, M., RAHTZ, S. y MITTELBACH, F.: *The L^AT_EX Graphics Companion. Illustrating Documents with T_EX and PostScript*, Addison-Wesley, 1997.
- [24] GOUALARD, F. y NEERGAARD, P. M.: «Making slides in L^AT_EX with prosper», documento distribuido con la clase, 2003.

- [25] GREEN, I.: «The citesort package», documento distribuido con el paquete, 1991.
- [26] GUNTERMANN, K.: « P^4 (PPower4) PDF Presentation Post Processor», documento distribuido con el programa, 1999.
- [27] GURARI, E.: «(L) $\hat{\text{A}}$ T_EX and T_EX for hypertext.» documento distribuido con el programa, 2003.
- [28] HANSEN, T.: «The bibunits package», documento distribuido con el paquete, 1999.
- [29] HENLICH, T.: «The MarVoSym font package», documento distribuido con el paquete, 2000.
- [30] JONES, D. M.: «A new implementation of L_AT_EX's indexing commands», documento distribuido con el paquete index, 1995.
- [31] KEMPSON, N. y AGUILAR-SIERRA, A.: «An 8-bit Implementation of BIBT_EX 0.99 with a Very Large Capacity», documento distribuido con el programa, 1996.
- [32] KNUTH, D. E.: *The T_EXbook*, Addison-Wesley, 1986.
- [33] _____ *The METAFONT book*, Addison-Wesley, 1986.
- [34] _____ *Digital tipography*, vol. 78 de *CSLI Lecture Notes*, Stanford University, 1999.
- [35] KUPKA, H. y DALY, P.: *A Guide to LaTeX2e, Document Preparation for Beginners and Users*, Addison-Wesley, 3rd ed edición, 1999.
- [36] LAMPORT, L.: «Makeindex: An index processor for L_AT_EX», documento distribuido con el programa MAKEINDEX, 1987.
- [37] _____ *L_AT_EX — A document Preparation System. User's guide and manual references*, Addison-Wesley, 2nd edición, 1994.
- [38] LEICHTER, J.: «The multirow package», documento distribuido con el paquete, 1994.
- [39] VAN LEUNEN, M. C.: *A Handbook for Scholars*, Oxford Univ. Press, 1992.
- [40] LEVY, S. y MURPHY, T.: «Using greek fonts with (L) $\hat{\text{A}}$ T_EX», documento distribuido con el paquete Igreek, 1996.
- [41] MARANGET, L.: «Hevea user documentation», documento distribuido con el programa, 2002.
- [42] MARTÍN DOMINGO, A.: «Xindy: Un sistema alternativo de creación y manejo de índices», en «EGUTH'99 Primer encuentro del grupo de usuarios de T_EX Hispanohablantes.» Universidad Politécnica de Madrid. I.C.E. y Dpto. de Física Aplicada a la Ingeniería E.T.S.I. Industriales, 1999.
- [43] MAYER, S.: «TEX converter», documento distribuido con el programa, 1999.
- [44] MCPHERSON, K.: «Displaying page layout variables», (modificado para L_AT_EX-2 ε por J.Braams), documento distribuido con el paquete layout, 1998.
- [45] MEHLICH, M.: «The fp package», documento distribuido con el paquete, 1995.
- [46] MITTELBACH, F.: «An environment for multicolumn output», documento distribuido con el paquete multicol, 1995.
- [47] _____ «A new implementation of L_AT_EX's tabular and array environment», documento distribuido con el paquete array, 1995.
- [48] NIEPRASCHK, R.: «The ps4pdf package», documento distribuido con el paquete, 2003.
- [49] OBERDIEK, H.: «The epstopdf package», documento distribuido con el paquete, 2001.
- [50] VAN OOSTRUM, P.: «Page layout in L_AT_EX», documento distribuido con el paquete fancyhdr, 1996.
- [51] ORLANDINI, M. y HAILPERIN, M.: «The multicolpar package», documento distribuido con el paquete, 1993.
- [52] PAGE, S.: «The doublespace package», documento distribuido con el paquete, 1986.
- [53] PATASHINK, O.: «BIBT_EXing», documento distribuido con el programa, 1988.
- [54] _____ «Designing BIBT_EX styles», documento distribuido con el programa, 1988.
- [55] POPINEAU, F.: «Affichez vos documents (L) $\hat{\text{A}}$ T_EX sur le web avex TEX4ht», *Cahiers GUTenberg* 37–38 (2000), 5–43.
- [56] RAHTZ, S. y BARROCA, L.: «The rotating package», documento distribuido con el paquete, 1995.
- [57] RAHTZ, S. y OBERDIEK, H.: «The hyperref package», documento distribuido con el paquete, 1999.

- [58] SMITH, C.: «The pageframe document style option», documento distribuido con el paquete, 1992.
- [59] SOMMERFELDT, A.: «The caption2 package», documento distribuido con el paquete, 2002.
- [60] MARTÍNEZ DE SOUSA, J.: *Diccionario de ortografía técnica*, Publicaciones de la Fundación Germán Sánchez Ruipérez, Madrid, 1990.
- [61] _____ *Diccionario de tipografía y del libro*, Paraninfo, 4^a edición, 1995.
- [62] SPIVAK, M. D.: *The Joy of TeX. A Gourmet Guide to Typesetting with the AMS-TeX macro package*, American Mathematical Society, 1990.
- [63] STORY, D. P.: «The web and exerquiz packages. Manual of usage», documento distribuido con el paquete, 2002.
- [64] SWIFT, M.: «The achicago BIBTeX style», documento distribuido con el paquete, 1998.
- [65] _____ «The achicago (L)ATeX package», documento distribuido con el paquete, 2001.
- [66] THORUP, K. K. y JENSEN, F.: «The calc package», documento distribuido con el paquete, 1995.
- [67] VÄTH, M.: «The varindex package», documento distribuido con el paquete, 2001.
- [68] VAN ZANDT, T.: «Documentation for fancybox.sty: Box tips and tricks for LATEX», documento distribuido con el paquete, 1993.
- [69] _____ «PSTricks: PostScript macros for generic TeX», documento distribuido con el paquete, 1993.

