
$$\int_{-1}^1 \phi_i \left[ 2\psi \delta_{lk} - \frac{\chi}{r^2} \mathbf{R} \right]$$

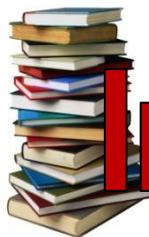
# CÓMO CREAR DOCUMENTOS CIENTÍFICOS DE CALIDAD CON HERRAMIENTAS DE SOFTWARE LIBRE

*Breve introducción a L<sup>A</sup>T<sub>E</sub>X, gnuplot y subversion*



Luis Alberto Padrón Hernández

**Subido por:**



# Interfase IQ

**Libros de Ingeniería Química y más**



<https://www.facebook.com/pages/Interfase-IQ/146073555478947?ref=bookmarks>

**Si te gusta este libro y tienes la posibilidad,  
cómpralo para apoyar al autor.**

# Cómo crear documentos científicos de calidad con herramientas de software libre

*BREVE INTRODUCCIÓN A LATEX,  
GNUPLOT Y SUBVERSION*

Luis Alberto Padrón Hernández



2011



PADRÓN HERNÁNDEZ, Luis Alberto

Cómo crear documentos científicos de calidad con herramientas de software libre [Recurso electrónico] : breve introducción a LaTeX, Gnuplot y Subversion / Luis Alberto Padrón Hernández. – Las Palmas de Gran Canaria : Universidad de Las Palmas de Gran Canaria, Servicio de Publicaciones, 2011

1 archivo PDF (10,1 MB, 109 p.)

ISBN 978-84-15424-15-4

1. Software libre. 2. LaTeX (Programa de ordenador) 3. Gnuplot (Programa de ordenador) 4. Subversion (Programa de ordenador) I. Universidad de Las Palmas de Gran Canaria, ed. II. Título.

004.4'232

*Cómo crear documentos científicos de calidad con herramientas de software libre.*

*Breve introducción a LaTeX, Gnuplot y Subversión*

La publicación de esta obra ha sido aprobada, tras recibir dictamen favorable en un proceso de evaluación interno, por el Consejo Editorial del Servicio de Publicaciones y Difusión Científica de la ULPGC.

© del texto: Luis Alberto Padrón Hernández

© de la edición: Universidad de Las Palmas de Gran Canaria

Servicio de Publicaciones y Difusión Científica

[serpubli@ulpgc.es](mailto:serpubli@ulpgc.es)

Producción: Servicio de Publicaciones y Difusión Científica de la ULPGC

1ª edición [versión electrónica], 2011

ISBN: 978-84-15424-15-4

Depósito Legal: GC 743-2011



Distribuidor de información  
del libro español en venta



Esta editorial es miembro de  
la UNE, lo que garantiza la  
difusión y comercialización  
de sus publicaciones a nivel  
nacional e internacional

Reservados todos los derechos por la legislación en materia de Propiedad intelectual. Ni la totalidad ni parte de esta obra, incluido el diseño de la cubierta, puede reproducirse, almacenarse o transmitirse en manera alguna por medio ya sea electrónico, químico, óptico, informático, de grabación o de fotocopia, sin permiso previo por escrito de la editorial.

# Índice general

<b>1. Introducción</b>	<b>8</b>
1.1. Motivación . . . . .	9
1.2. Sobre las distintas herramientas de elaboración de documentos científico-técnicos . . . . .	9
1.2.1. Una hipotética herramienta ideal . . . . .	9
1.2.2. L <sup>A</sup> T <sub>E</sub> X: Una buena opción . . . . .	11
1.2.3. Sobre T <sub>E</sub> X y L <sup>A</sup> T <sub>E</sub> X . . . . .	14
1.3. Sobre las gráficas e ilustraciones en documentos científico-técnicos . . . . .	15
1.4. Sobre el contenido . . . . .	16
1.5. Todo el software presentado y utilizado en este manual es software libre . . . . .	17
<b>2. Producción de documentos científico-técnicos de alta calidad con L<sup>A</sup>T<sub>E</sub>X</b>	<b>18</b>
2.1. Introducción . . . . .	19
2.2. Puesta a punto . . . . .	19
2.2.1. Obtención e instalación de un sistema L <sup>A</sup> T <sub>E</sub> X . . . . .	19
2.2.2. Obtención e instalación de T <sub>E</sub> XMaker . . . . .	20
2.2.3. Nuestro primer documento L <sup>A</sup> T <sub>E</sub> X . . . . .	21
2.2.4. Obtención de documentos en formato .pdf . . . . .	22
2.2.5. Los errores en L <sup>A</sup> T <sub>E</sub> X . . . . .	22
2.2.5.1. Ejemplos de errores en L <sup>A</sup> T <sub>E</sub> X . . . . .	23
2.2.5.2. Propagación de errores . . . . .	23
2.3. Estructura y contenido de un documento L <sup>A</sup> T <sub>E</sub> X . . . . .	23
2.3.1. El preámbulo . . . . .	24
2.3.2. Los paquetes . . . . .	26
2.3.3. Los entornos . . . . .	26
2.3.4. Los comentarios . . . . .	27
2.3.5. Nuevos párrafos y líneas . . . . .	28
2.3.6. Capítulos, secciones, subsecciones, . . . . .	28
2.4. Para empezar a trabajar con L <sup>A</sup> T <sub>E</sub> X . . . . .	30

2.4.1. Modificando la tipografía: familias, perfiles y tamaños de letra .....	30
2.4.2. Incluyendo gráficos e ilustraciones .....	30
2.4.2.1. Posicionamiento y etiquetado automático de imágenes como objetos flotantes .....	32
2.4.3. Etiquetando .....	34
2.4.4. Haciendo listas .....	35
2.4.5. Incluyendo tablas .....	36
2.4.5.1. Tablas como objetos flotantes .....	37
2.4.6. Incluyendo símbolos y fórmulas matemáticas .....	38
2.4.6.1. El modo matemático y sus entornos .....	38
2.4.6.2. Algunos conceptos importantes .....	39
2.4.6.3. Los símbolos en $\text{\LaTeX}$ .....	40
2.4.6.4. Modos de texto dentro del modo matemático .....	41
2.4.6.5. Algunos elementos importantes en modo matemático .....	41
2.4.6.6. Espacios y puntos en modo matemático ..	42
2.4.6.7. Matrices y vectores .....	43
2.4.6.8. Paréntesis de tamaño variable .....	44
2.4.6.9. Referencias a ecuaciones .....	45
2.4.7. Escribiendo documentos grandes .....	46
2.4.7.1. El comando <code>input</code> .....	46
2.4.7.2. El comando <code>include</code> .....	46
2.4.7.3. A tener en cuenta a la hora de compilar..	48
2.5. Y mucho más .....	48
<b>3. Gestión de la bibliografía en <math>\text{\LaTeX}</math>: <i>BiBTeX + JabRef</i></b>	<b>49</b>
3.1. Introducción .....	50
3.2. La bibliografía <i>a mano</i> : El entorno <code>thebibliography</code> .....	50
<b>Bibliografía</b>	<b>51</b>
3.3. Automatizar la bibliografía .....	52
3.3.1. Obtención e instalación de <i>JabRef</i> .....	53
3.3.2. Creación de archivo .bib con <i>JabRef</i> .....	53
3.3.2.1. Sobre el formato del campo 'autores' y otros aspectos a tener en cuenta .....	55
3.3.3. Generación de la bibliografía con <i>BiBTeX</i> .....	55
<b>4. Representación gráfica de datos y funciones con <i>gnuplot</i></b>	<b>57</b>
4.1. ¿Qué es <i>gnuplot</i> y para qué sirve? .....	58
4.2. Obtención, instalación y ejecución de <i>gnuplot</i> .....	61
4.3. Representación de funciones analíticas .....	61

4.4. Representación de ficheros de datos .....	62
4.5. Transformación de datos .....	64
4.6. Exportación de gráficos: elección de formato de imagen y de fichero de salida .....	65
4.7. Aprovechando el trabajo: los ficheros de procedimiento ..	66
4.8. Títulos, etiquetas, leyendas y rangos .....	67
4.9. Abreviaturas .....	69
4.10. Utilización de estilos de líneas y puntos .....	69
4.11. Introducción de símbolos en <i>gnuplot</i> .....	73
4.12. Gráficos múltiples .....	75
4.12.1. Utilizando la opción <i>layout</i> .....	76
4.12.2. Gráficos múltiples sin la opción <i>layout</i> .....	79
4.13. Funciones implementadas en <i>gnuplot</i> .....	82
4.14. Operadores y condicionales en <i>gnuplot</i> .....	82
4.14.1. Operadores de dos argumentos .....	82
4.14.2. Operadores de un único argumento .....	84
4.14.3. Condicionales .....	84
4.15. <i>Gnuplot</i> y <i>LATEX</i> .....	85
4.15.1. Haciendo que sea <i>LATEX</i> quien procese el texto y los símbolos de una imagen: el terminal <i>epslatex</i> ..	85
4.15.2. <i>Gnuplot</i> , <i>beamer</i> y fondos transparentes en una ilustración .....	86
<b>5. Control de versiones con <i>subversion</i></b> .....	<b>89</b>
5.1. ¿Qué es y para qué sirve un sistema de control de versiones? .....	90
5.2. <i>Subversion</i> : un sistema de control de versiones .....	90
5.3. ¿Interferirá <i>subversion</i> en mi trabajo diario? .....	91
5.4. Obtención e instalación de <i>subversion</i> .....	91
5.5. Conceptos básicos y flujo de trabajo .....	92
5.5.1. Creación de un repositorio .....	94
5.5.2. Sobre la estructura recomendada de un repositorio: <i>tags</i> y <i>branches</i> .....	95
5.5.3. Obtención de una copia local de trabajo .....	96
5.5.4. Información de la copia local de trabajo .....	97
5.5.5. Trabajo: creación, edición, copia y eliminación de archivos y carpetas .....	97
5.5.6. Envío de los cambios al repositorio .....	98
5.5.7. Abreviaturas de comandos y opciones .....	99
5.5.8. Ayuda y documentación de <i>subversion</i> .....	99
5.5.9. Concepto de revisión .....	99
5.5.9.1. Revisiones Head y Base .....	100
5.5.10. Actualización de contenidos .....	100

*Índice*

5.5.11. Conocimiento del estado actual de archivos y carpetas .....	100
5.5.12. Conocimiento del estado previo de archivos y carpetas .....	102
5.5.12.1. Obteniendo un listado de revisiones .....	102
5.5.12.2. Obteniendo diferencias entre distintos instantes .....	103
5.5.12.3. Recuperando un archivo de una revisión anterior .....	104
5.5.12.4. Recuperando un archivo de la revisión actual .....	104
5.5.12.5. Deshacer cambios enviados anteriormente .....	105
5.5.13. Conflictos .....	105
5.5.14. Creación de tags y branches .....	106
5.6. Repositorios en red .....	107
<b>Bibliografía</b>	<b>109</b>

# **CAPÍTULO 1**

## **Introducción**

## **1.1. Motivación**

Este manual pretende servir de apoyo a toda persona que deba enfrentarse, ya sea de manera esporádica o regular, a la tarea no trivial de producir un documento científico-técnico en su conjunto. La intención es que investigadores, profesores, estudiantes y profesionales puedan sacar partido de lo que aquí se presenta.

Este manual no pretende, sin embargo, ser una obra de referencia exhaustiva de las herramientas que aquí se van a tratar, ya que existen fuentes de información específicas y muy completas para todos los casos. Por contra, sus objetivos son: (a) mostrar al lector las ventajas asociadas con el uso de herramientas como las aquí propuestas, (b) constituir un primer paso *indoloro*, rápido y eficaz para introducirse a la utilización de dichas herramientas y estar en posición de acudir a otras fuentes para la consulta de dudas concretas, y (c) satisfacer de forma coherente e interconectada las necesidades básicas que asaltan a quien pretende escribir un documento científico-técnico, que no se restringen (generalmente) a la redacción y presentación del texto, sino que se extienden a la generación de gráficos e ilustraciones, la gestión de bibliografía y la gestión del trabajo individual y/o colectivo.

Plasmar en un documento de carácter científico-técnico aquello que se desea transmitir, junto a los datos que apoyan la tesis propuesta y/o las ilustraciones que hagan más sencilla la compresión de lo que se pretende contar, no es tarea fácil. Y lo que es peor, en muchas ocasiones este proceso se ve coartado por las capacidades de las herramientas utilizadas, lo que, en ocasiones, produce un resultado no del todo satisfactorio. Este manual pretende mostrar algunas herramientas alternativas que permiten, precisamente, la obtención de resultados satisfactorios y de alta calidad, con un coste asequible en términos de tiempo, esfuerzo y dinero.

## **1.2. Sobre las distintas herramientas de elaboración de documentos científico-técnicos**

### **1.2.1. Una hipotética herramienta ideal**

Durante mucho tiempo pensé que, el que ahora es el sistema operativo más popular del mundo, constituía la única forma de hacer funcionar un ordenador, cuando lo cierto es que sí existen otras alternativas. Algo parecido ocurre con la forma de producir de documentos, dado que la costumbre general de utilizar una herramienta concreta para la preparación y edición de documentos (sí, justamente el programa que usted está pensando) da a pensar que no existen mejores alternativas.

El estilo de los procesadores de texto de uso general se ajustan a lo que usualmente se conoce como *wysiwyg* (*what you see is what you get*). En esencia, el objetivo es mostrar en pantalla y *en tiempo real* el documento que está siendo creado de la manera más cercana posible a cómo será impreso posteriormente. Pero . . . ¿es ésta la mejor forma de hacerlo? ¿Qué características tendría el método ideal para producir documentos científico-técnicos utilizando el ordenador?

Podríamos plantearnos que un hipotético método o herramienta ideal tendría, entre otras, las siguientes características:

- **Permitiría, facilitaría y promovería que el escritor se preocupase principalmente de lo más importante: el contenido y su organización.** En las herramientas tipo *wysiwyg*, el autor debe enfrentarse simultáneamente a dos tareas. Por un lado, debe redactar y organizar las ideas que quiere transmitir (lo realmente importante) y, al mismo tiempo, debe preocuparse de aspectos relacionados con la tipografía y la composición del texto: justificación, tamaños, fuentes, estilos de los distintos tipos de cabecera (de capítulo, de sección, de subsección, de pie de figura, de nota al pie, bibliografía, citas y referencias, etc.), posición y características de las figuras o ilustraciones, homogeneidad en la presentación de los distintos tipos de elementos, etc. Esta última parte del trabajo, relacionada con la tipografía y la composición del texto, no debería mezclarse con la elaboración del contenido en sí. De hecho, la mezcla de ambos momentos distrae al autor de su verdadero cometido y, como consecuencia de ello, disminuye su productividad.
- **Generaría un resultado tipográfico y compositivo de alta calidad,** incluso asistiendo al autor en la toma de decisiones a la hora de dónde y cómo colocar elementos tales como figuras o ilustraciones siguiendo criterios compositivos pre establecidos o, en su caso, definidos por el propio usuario.
- **Facilitaría la generación de un único documento a partir de distintos ficheros** si fuera necesario y/o la complejidad del documento así lo sugiriese. Es decir, facilitaría la integración de distintos archivos 'sencillos' en un documento maestro 'complejo'. Piénsese que, por lo general, se asocia la idea de un fichero con un documento o, en caso de partir un documento en distintos archivos (por capítulos, por ejemplo) es generalmente el autor el que tiene que preocuparse por conseguir un paginado coherente, un índice general, una homogeneidad en todo el documento, etc. Una herramienta ideal permitiría esta integración de forma natural, y además cuidaría por nosotros de mantener la coherencia del documento final. Es más, muchos de esos archivos podrían contener básicamente el texto que conforma el documento, pero podría haber otros tipos de archivos que contuviesen las ilustraciones, la bibliografía, o incluso las *instrucciones* para formatear y ordenar el documento en su conjunto, de manera que modificar dichas *instrucciones* pudiera limitarse a modificar o sustituir di-

chos ficheros sin tener que realizar cambios en los ficheros relacionados con el contenido en sí.

- Sería capaz de manejar documentos grandes o con muchas figuras con la misma facilidad y fluidez con la que maneja un documento pequeño, idea íntimamente relacionada con la nombrada en el punto anterior.
- Gestionaría autónomamente la bibliografía y las referencias a elementos tales como tablas, figuras o ecuaciones. Piénsese que durante la elaboración de un documento complejo, el autor irá añadiendo, eliminando y modificando elementos como tablas, figuras y ecuaciones que llevan numeración asociada y que son referenciados en múltiples ocasiones desde el texto en sí. Durante el proceso, el orden de los elementos y, por tanto, su numeración, puede ir cambiando, lo que implica que el autor debe estar atento a este tipo de detalles. Lo mismo se aplica a la numeración de los apartados o subapartados de cada capítulo y, de manera diferente, a la bibliografía. También al índice o índices del documento completo. Parece claro que la herramienta ideal debería gestionar automáticamente toda esta información, numerando y referenciando de forma autónoma todos los elementos del documento.
- El trabajo generado sería portable, independiente del sistema operativo, del programa y de la máquina con el que fue generado. Piénsese que cuando esto no es así, el autor deja de tener control sobre su propio trabajo en cuanto que su documento queda vinculado a un soporte determinado. ¿Qué ocurrirá si dentro de una decena de años queremos editar de nuevo nuestro trabajo y ya no tenemos el mismo software a mano? ¿Y si lo queremos compartir con alguien que no tiene acceso a él?
- No necesitaría ordenadores potentes para crear un documento.

### **1.2.2. *LATeX: Una buena opción***

Muchos de los deseos arriba formulados pueden ser satisfechos en gran medida a través del uso de *LATeX*, un sistema avanzado de creación de documentos con ayuda del ordenador.

Un documento *LATeX*, por complejo que sea, por muchas imágenes y ecuaciones que tenga, se redacta en un fichero de texto plano creado utilizando cualquier editor de textos básico: el 'bloc de notas' o el 'edit' de MS Windows, el 'vi', el 'emacs' o el 'gedit' de GNU/Linux, o el 'TextEdit' de Mac OS X. *Texto plano* se refiere a un fichero que no contiene ningún tipo de formato (justificación, negrita, cursiva, etc.) de las que aplican los programas wysiwyg, y que tan sólo contiene 95 caracteres básicos definidos en el estándar ASCII y que representan básicamente las letras mayúsculas y minúsculas, los números del 0 al 9, los signos de puntuación. Este documento escrito con cualquier editor de texto será luego procesado por un programa capaz de interpretar el documento y que

```
% Aquí comienza la cabecera del documento
\documentclass[a4paper,12pt]{article}
\usepackage[utf8x]{inputenc}
\usepackage[spanish]{babel}
\title{Esto es un documento tipo Artículo}
\author{Luis A. Padrón}
% Aquí termina la cabecera del documento

% Inicio del contenido del documento
\begin{document}
\maketitle
```

Este es un documento de demostración. Como puede verse, está compuesto, básicamente, de cabecera y cuerpo del documento. En la cabecera se indica tamaño de papel, tamaño de fuente, tipo de documento, título, autor, y el uso de dos paquetes: el primero permite utilizar letras acentuadas y otros caracteres (no incluidos inicialmente en el código ASCII) y el segundo configura el idioma del documento a español. Durante el cuerpo del documento se le da la instrucción 'Imprime el título' (`\textit{\maketitle}`) y a continuación se introduce el contenido principal del documento.

Como puede verse, las instrucciones (anotaciones) se dan en inglés, pero las más comunes son, en general, fáciles de interpretar y recordar. % Fin del documento  
`\end{document}`

## Esto es un documento tipo Artículo

Luis A. Padrón

4 de junio de 2010

Este es un documento de demostración. Como puede verse, está compuesto, básicamente, de cabecera y cuerpo del documento. En la cabecera se indica tamaño de papel, tamaño de fuente, tipo de documento, título, autor, y el uso de dos paquetes: el primero permite utilizar letras acentuadas y otros caracteres (no incluidos inicialmente en el código ASCII) y el segundo configura el idioma del documento a español. Durante el cuerpo del documento se le da la instrucción 'Imprime el título' (`\maketitle`) y a continuación se introduce el contenido principal del documento.

Como puede verse, las instrucciones (anotaciones) se dan en inglés, pero las más comunes son, en general, fáciles de interpretar y recordar.

**Figura 1.1: Ejemplo de documento tipo artículo. Arriba fichero tipo L<sup>A</sup>T<sub>E</sub>X, formado por la cabecera y el cuerpo. Abajo, resultado tras ser procesado por el programa L<sup>A</sup>T<sub>E</sub>X**

generará la versión imprimible del mismo. En los siguientes capítulos se entrará en detalle sobre esto. En la figura 1.1 puede verse un ejemplo de documento  $\text{\LaTeX}$  en texto plano y su resultados una vez procesado.

Crear un documento usando  $\text{\LaTeX}$  puede asociarse al proceso *clásico* de creación de libros. El proceso empezaba, obviamente, en la redacción del *manuscrito* por parte del autor. Después de revisar y corregir el manuscrito, el autor realizaba una serie de anotaciones sobre el mismo indicando al impresor, por ejemplo, qué texto debe aparecer en negrita o en cursiva, dónde comienzan y terminan los capítulos o cuáles son las ilustraciones de una determinada sección. Sin embargo, era el impresor el que, siendo experto en composición y edición de documentos, se encargaba de componer las planchas que servirían finalmente para imprimir la versión final de la obra.

Cuando trabajamos en  $\text{\LaTeX}$  podemos pensar que estamos generando un manuscrito sobre el que realizamos las anotaciones necesarias para que el impresor (que en este caso deja de ser una persona para pasar a ser un programa:  $\text{\TeX}$ ) pueda interpretarlas y componer, siguiendo los criterios adecuados, un documento de la mayor calidad. Para entendernos, si queremos que *este texto* aparezca en cursiva en su versión imprimible, sólo tendremos que introducir, en el fichero generado en texto plano, una anotación estándar, en este caso `\textit{este texto}`, junto con una indicación de la cantidad de texto a la que afecta, delimitándolo con las llaves `{ }` . En resumen, `\textit{este texto}` producirá el resultado deseado.

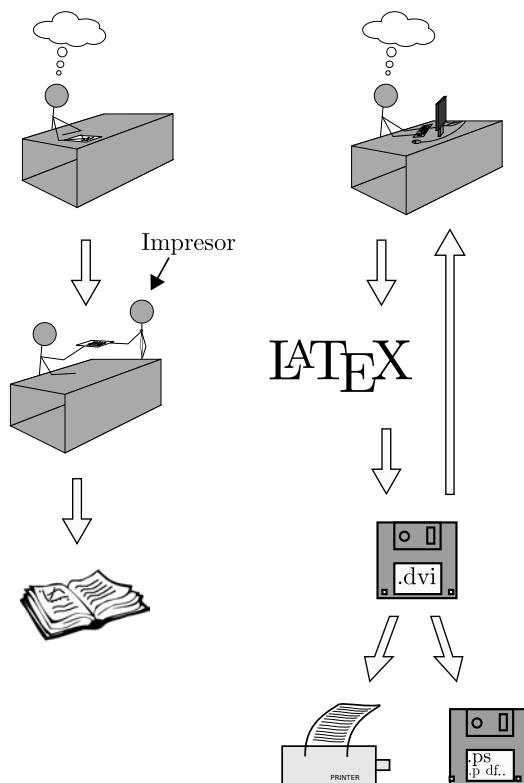


Figura 1.2: Proceso de creación de documentos

El proceso y la idea de fondo se ilustra de forma resumida en la figura 1.2, donde vemos el paralelismo entre los binomios autor-impresor y autor- $\text{\LaTeX}$ . La figura ilustra también cómo, cuando  $\text{\LaTeX}$  procesa nuestro documento de entrada, da como resultado un documento visible en la pantalla del ordenador e independiente del dispositivo utilizado en formato dvi (device independent) o,

en su caso, un registro de los errores que pudieran haberse cometido a la hora de *anotar* el contenido del documento. Esta versión imprimible del documento puede ser utilizada para refinar el resultado final, para imprimir directamente, o para convertirlo a otros formatos de archivo.

### **1.2.3. Sobre *TeX* y *LATEX***

*LATEX* fue desarrollado inicialmente por Leslie Lamport a principios de los años 80 [1] como un intérprete de alto nivel para facilitar la comunicación del usuario con *TeX*, un lenguaje y motor de composición de textos de bajo nivel, extremadamente potente y versátil de cara a la generación de documentos de altísima calidad tipográfica (pero relativamente complejo y laborioso de usar) creado previamente por Donald Knuth [2]. *LATEX* es, en esencia, un conjunto de macros escritas en lenguaje *TeX* para la realización de multitud de tareas que permite al autor concentrarse principalmente en el contenido y la estructura lógica del documento. Su concepto y estructura fomenta además su desarrollo progresivo con la incorporación de nuevas posibilidades, desarrollo que continua hasta el día de hoy. (Más detalles sobre su historia pueden leerse en [3].)

¿Y cómo es que *LATEX* es capaz de ofrecer documentos de calidad tipográfica muy superior a la ofrecida por herramientas *wysisyg*? *LATEX* utiliza una serie de algoritmos para calcular cuál es la posición óptima de cada uno de los elementos que componen un documento, para lo cual trabaja sobre un fichero que contiene el texto anotado del documento completo. De este modo, *LATEX* tiene en cuenta todos los elementos del documento para su composición, al contrario de lo que ocurre con las herramientas *wysisyg*, que necesitan mostrar en pantalla, de manera casi instantánea, la vista de página. Además, *TeX* trabaja con el concepto de cajas (refiriéndose al espacio ocupado por cada elemento) y de pegamento (como la sustancia que relaciona unas cajas con otras y que, según sus propiedades elásticas y según los criterios del impresor, gobierna cómo se modifican los espacios entre elementos para conseguir una composición lo más correcta posible). Estos elementos son localizados y relacionados por el algoritmo de manera iterativa hasta obtener la distribución óptima bajo unas reglas determinadas fijadas inicialmente por el maquetador, pero que pueden ser modificadas por el usuario. Dichas reglas establecen, por ejemplo, cuáles son los espacios que hay que dejar tras un punto y seguido o tras un punto y coma; cuánto se puede *estirar* o *comprimir* una línea para conseguir el justificado del texto; cuál es el porcentaje máximo de una página que puede estar ocupado por figuras y dónde deben ir éstas preferentemente colocadas; o cuánto espacio en blanco puede dejarse en una página y cómo debe reorganizarse el texto para minimizarlo.

### **1.3. Sobre las gráficas e ilustraciones en documentos científico-técnicos**

Por lo general, los documentos de carácter científico-técnico vienen acompañados de gráficas y/o ilustraciones, cuya elaboración y presentación no siempre se realizan de la forma más adecuada. En muchas ocasiones, las representaciones gráficas o las ilustraciones no son de buena calidad. En otros casos, aún siendo la imagen de buena calidad, no se ha cuidado que sus características sean coherentes con el resto de ilustraciones y con el propio texto.

Gran parte de este problema puede solucionarse siendo consciente de su existencia y esforzándose en producir un documento cuyos elementos estén todos presentados de forma coherente en cuanto a tipología, colores, tipos de líneas o tamaños y tipos de letra. Por otro lado, para obtener imágenes de calidad adecuada a cada aplicación es útil saber que hay dos grandes tipos de imágenes, las imágenes tipo *mapa de bits* y las *imágenes vectoriales*.

Las imágenes tipo *mapa de bits* son posiblemente las más comunes, con extensiones .bmp, .jpg, .gif o .tiff entre otras. En general, estos formatos almacenan la información de la imagen píxel a píxel, como cuando se saca una fotografía. La gran mayoría utiliza algoritmos de compresión e interpolación de la información de la imagen con el objetivo principal de disminuir el tamaño del archivo. Las imágenes de este tipo serán dependientes de la resolución con la que fueron creadas, de modo que al intentar ampliar una zona de la misma se perderá nitidez con mucha rapidez. Aún así, son el tipo idóneo para almacenar imágenes de tipo fotográfico.

Por contra, las *imágenes vectoriales* no almacenan la información píxel a píxel, sino que almacenan la información de los objetos que definen la imagen, de tal modo que el ordenador pueda posteriormente presentar la imagen. Aunque existen diversos formatos vectoriales asociados a distintos programas comerciales, los tres formatos principales, generables y modificables con multitud de programas distintos son: .ps (postscript), .eps (encapsulated postscript) y .svg (scalable vector graphics). Los objetos almacenados (líneas, áreas, círculos, polígonos, caracteres, ...) quedan descritos por ecuaciones matemáticas y por sus propiedades y características (tipo, posición o puntos que lo definen, rotación, prioridad respecto de otros objetos, transparencias, color, grosor, ...). Por todo esto, este tipo de imagen presenta dos ventajas principales: (a) puede ser ampliada de forma indefinida sin perder calidad, dado que es el propio ordenador el que la reinterpreta en cada momento, y (b) cada objeto puede ser modificado posteriormente de forma independiente a los demás. En el caso que nos ocupa, donde la mayoría de las imágenes a manejar serán ilustraciones y representaciones gráficas, este tipo de imagen presenta la ventaja adicional de su pequeño tamaño que, junto con su calidad y flexibilidad, representan la solución ideal en documentos científico-técnicos.

El capítulo 4 de este manual nos introducirá, con suficiente detalle, a una herramienta específica para la creación de representaciones gráficas vectoriales: *gnuplot*. Por contra, la generación de ilustraciones queda, por su amplitud, fuera del ámbito de este manual. Sin embargo, en cuanto a ilustraciones vectoriales aptas para documentos científico-técnicos, *InkScape* es un programa abierto, muy recomendable y con una interfaz amigable. Puede descargarse libremente desde <http://www.inkscape.org/> e incluye diversos tutoriales. Después de crear una ilustración, *InkScape* permite exportar la imagen a multitud de formatos útiles para su inclusión en un documento, como por ejemplo los de tipo .eps o .ps. Además, la versión 0.48 de *InkScape* incluye una funcionalidad especial a la hora de insertar ilustraciones en documentos *LATEX*, tal y como se explica en [4].

#### 1.4. Sobre el contenido

El siguiente capítulo de este manual tratará de introducir al lector en la utilización de *LATEX* de manera amena y sencilla, y sólo a un nivel básico que permita al usuario crear sus primeros documentos y entender e interiorizar el funcionamiento del sistema de modo que pueda ser luego autónomo para acudir a otros recursos a resolver dudas concretas que le puedan surgir.

Pero un documento científico-técnico no está compuesto, generalmente, sólo de texto. El autor puede necesitar generar e introducir en el documento final otros elementos tales como ilustraciones y representaciones gráficas de datos y resultados, además de la bibliografía pertinente. Por esto, los capítulos 3 y 4 tratan la generación de bibliografía con *BiBTEX* y *JabRef*; y de representaciones gráficas con *gnuplot*.

Finalmente, el capítulo 5 trata un asunto al que todos tenemos que enfrentarnos, de una manera u otra, durante la creación de documentos o programas informáticos: el control de versiones. A primera vista puede parecer una cuestión baladí, sobre todo cuando se trata de pequeños proyectos, como la redacción de un artículo o de un proyecto fin de carrera. Pero ¿no cree usted que sería útil utilizar un sistema que, sin interferir con las herramientas que utilicemos para nuestro trabajo, y sin depender de ellas, pueda controlar y almacenar los distintos momentos por los que va pasando nuestro trabajo? Este tipo de sistemas, entre otras cosas, permite (a) llevar un registro del trabajo realizado, documentando cada paso en la medida que el usuario estime oportuna, (b) conservar 'copias congeladas' de los archivos y/o carpetas que configuran un trabajo a lo largo del tiempo de tal modo que podamos recuperar en cualquier momento un archivo determinado en el estado que tenía en una fecha y hora dadas, (c) conocer los cambios que ha sufrido cualquier archivo entre dos instantes de tiempo determinados, (d) generar variantes del trabajo manteniendo un cierto número de elementos comunes, evitando de este modo duplicidades

innecesarias, y (e) proporcionar una manera sencilla de trabajar de manera colaborativa sobre un mismo documento. Podemos encontrar todas estas características (y muchas más) en los diversos sistemas disponibles para control de versiones. De entre ellos, el capítulo 5 tratará de introducir al lector en el uso de uno de los más utilizados: *subversion*.

Todos los capítulos han sido intencionadamente redactados de manera escueta y concisa, intentando presentar la información necesaria en poco espacio para que usted pueda comenzar a utilizar las ideas e herramientas que se presentan a la mayor brevedad posible, sin tener que invertir demasiado tiempo leyendo este manual. Sin embargo, se presentan numerosos ejemplos con la intención de ilustrar lo que se intenta transmitir. Se anima al lector a que vaya reproduciendo y jugando con los ejemplos al mismo tiempo que lee el manual.

### **1.5. Todo el software presentado y utilizado en este manual es software libre**

Por último, todas las herramientas tratadas en este manual pueden ser consideradas como de **software libre**, lo que significa que todo usuario es libre de copiar y ejecutar el software, así como de distribuirlo y modificarlo con ciertas condiciones, según las especificidades de cada programa. También significa que los propios usuarios podemos participar en la elaboración y mejora de las herramientas y, aunque es menos importante, que los programas pueden ser, por lo general, conseguidos, utilizados y redistribuidos de manera gratuita y legal.

Además, todas las herramientas utilizadas en este manual pueden ser ejecutadas en cualquiera de los tres sistemas operativos mayoritarios:  GNU/Linux,  Mac OS X y  MS Windows. A pesar de que este manual y todo su material han sido generados íntegramente en  GNU/Linux, todo lo que aquí se presenta puede ser realizado perfectamente en cualquier sistema operativo. Las pocas diferencias concretas que existan serán puntualizadas en su momento de manera que cualquier usuario pueda hacer uso de todas estas herramientas.

**CAPÍTULO 2**

**Producción de documentos científico-técnicos de  
alta calidad con L<sup>A</sup>T<sub>E</sub>X**

## 2.1. Introducción

El capítulo 1 ya presentó y motivó el uso de L<sup>A</sup>T<sub>E</sub>X como una herramienta idónea en la producción de documentos científico-técnicos de alta calidad. La mayor barrera de todo usuario habituado a los editores de texto más utilizados (de tipo wysiwyg) es la necesidad de un cambio de mentalidad y de forma de trabajar. Por esta razón, en este capítulo se pretende introducir al lector en el manejo de L<sup>A</sup>T<sub>E</sub>X de manera sencilla, no con el objetivo de alcanzar el nivel de experto, sino únicamente de romper, de manera asequible, la barrera que supone el cambio desde herramientas wysiwyg.

Para conseguir este objetivo, se empezará describiendo brevemente el proceso de instalación del sistema L<sup>A</sup>T<sub>E</sub>X en cualquier sistema operativo, se presentará un programa llamado *TeXMaker* de apoyo a la edición de documentos L<sup>A</sup>T<sub>E</sub>X y se darán las nociones necesarias para que el usuario pueda comenzar a generar sus primeros documentos y sea capaz de solucionar el resto de dudas que le puedan ir surgiendo.

## 2.2. Puesta a punto

### 2.2.1. Obtención e instalación de un sistema L<sup>A</sup>T<sub>E</sub>X

Para comenzar es necesario instalar un sistema L<sup>A</sup>T<sub>E</sub>X. La tabla 2.1 resume el nombre y la dirección web de los sistemas L<sup>A</sup>T<sub>E</sub>X recomendados para cada uno de los tres sistemas operativos. Antes de continuar, el usuario debe instalar el sistema correspondiente a su caso particular.

En el caso de MS Windows y MiKTeX, es altamente recomendable instalar la versión básica en los ordenadores con conexión a Internet en lugar de la versión completa, de gran tamaño. Además, los usuarios de Windows deberán asegurarse de tener instalado un visor de documentos PDF (por ejemplo, adobe reader) y los programas Ghostscript y Gsview para procesar y visualizar ficheros PostScript. Éstos últimos, que deben instalarse en ese orden, pueden ser descargados desde la página <http://pages.cs.wisc.edu/~ghost/>.

Sistema Operativo	Sistema L <sup>A</sup> T <sub>E</sub> X	Dirección web
MS Windows	MiKTeX	<a href="http://miktex.org">http://miktex.org</a>
GNU/Linux	texlive	<a href="http://tug.org/texlive/">http://tug.org/texlive/</a>
Mac OS X	MacTeX	<a href="http://www.tug.org/mactex/">http://www.tug.org/mactex/</a>

Tabla 2.1: Sistema L<sup>A</sup>T<sub>E</sub>X recomendado en función del sistema operativo

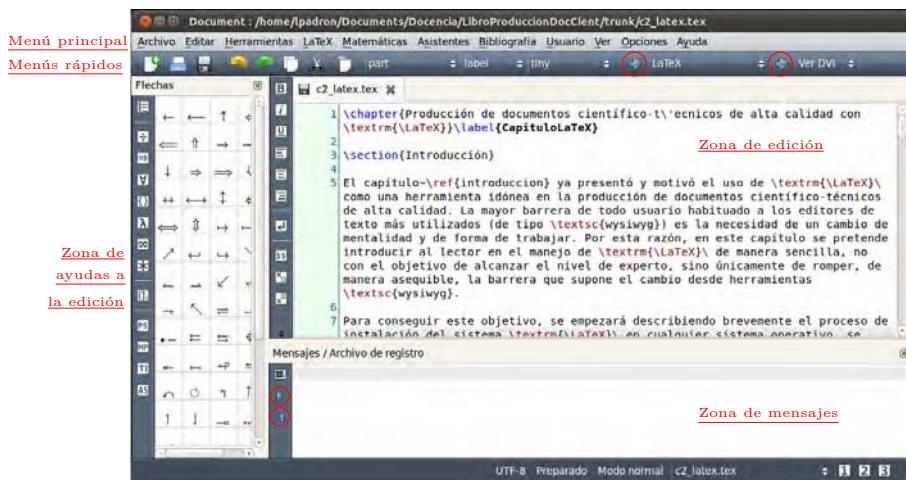


Figura 2.1: Pantallazo de la interfaz de *TeXMaker*

En el caso de los usuarios de  $\Delta$  GNU/Linux, hay que tener en cuenta que texlive está generalmente disponible en los repositorios oficiales de la gran mayoría de distribuciones.

### 2.2.2. Obtención e instalación de *TeXMaker*

Además del sistema  $\text{\LaTeX}$  descrito en el punto anterior, es altamente recomendable hacer uso de alguno de los entornos integrados de edición  $\text{\LaTeX}$  disponibles, que facilitan la redacción, edición, compilación y revisión de documentos. De entre todos ellos vamos a introducirnos a *TeXMaker*, abierto y disponible para los tres sistemas operativos<sup>1</sup>. *TeXMaker* puede ser descargado libremente desde su página web oficial: <http://www.xm1math.net/texmaker/>, si bien en el caso del usuario de  $\Delta$  GNU/Linux, el programa está generalmente disponible en los repositorios oficiales, por lo que no es necesario descargarlo. Para los usuarios de  $\blacksquare$  MS Windows o de  $\apple$  Mac OS X, el proceso de descarga e instalación es el habitual y no es complicado. La figura 2.1 muestra un pantallazo del programa.

Antes de empezar a trabajar, es recomendable cambiar algunas opciones de *TeXMaker* en Opciones → configurar Texmaker. A la izquierda de la ventana de configuración se pueden seleccionar tres bloques de opciones distintos: Comandos, Compilación rápida y Editor.

---

<sup>1</sup> Kile es también una excelente opción para los usuarios de  $\Delta$  GNU/Linux.

1. Para que  $\text{\LaTeX}$  y  $\text{\TeXMaker}$  traten correctamente las tildes y demás caracteres especiales utilizados en el español, debemos ir al bloque Editor y, en el desplegable Codificación del editor elegir una codificación adecuada. La opción UTF-8 es muy recomendable, aunque la ISO-8859-1 también es válida.
2. Únicamente en el caso de los usuarios de MS Windows,  $\text{\TeXMaker}$  suele configurar erróneamente los visores de archivos DVI y PS (es decir, los programas ghostscript y gsview descritos en el apartado anterior). Para configurarlos correctamente, y en el bloque Comandos, deben corregirse las siguientes dos entradas:
  - a) En Visor DVI debe aparecer: yap %.dvi
  - b) En Visor PS debe aparecer: gsview32 %.ps

### 2.2.3. Nuestro primer documento $\text{\LaTeX}$

Vamos a crear nuestro primer documento, para comprobar la instalación de  $\text{\LaTeX}$  y para ir familiarizándonos con él.

El primer paso debe ser crear un documento nuevo en blanco haciendo *clic* en Archivo → Nuevo en el menú principal. Introduzca ahora el siguiente texto de ejemplo (que en breve explicaremos):

```
\documentclass[a4paper,12pt]{report}
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}

\begin{document}
Este es mi primer documento \LaTeX.
\end{document}
```



Herramientas	LaTeX	Matemát
Compilación rápida	F1	
LaTeX	F2	
Ver DVI	F3	
Dvi->PS	F4	
Ver PS	F5	
PDFLaTeX	F6	
Ver PDF	F7	
PS->PDF	F8	
DVI->PDF	F9	

Guarde el documento con un nombre de su elección, por ejemplo *ejemplo-1.tex*, y a continuación compile el documento haciendo *clic* en Herramientas → LaTeX en el menú principal (si el programa le devuelve algún error, revise que el texto esté introducido exactamente como se muestra arriba y vea la sección 2.2.5). Finalmente, para ver el resultado, vaya a Herramientas → Ver DVI, tras lo que debería abrirse otra ventana con su primer documento, compuesto por una página con la frase 'Este es mi primer documento  $\text{\LaTeX}$ '. Estos dos últimos pasos pueden realizarse también pulsando sobre las flechas azules de

la parte derecha de los menús rápidos (véanse los círculos rojos en la parte superior de la figura 2.1) tras seleccionar `LaTeX` y `Ver DVI` en los desplegables, o también pulsando las teclas F2 y posteriormente F3.

#### 2.2.4. Obtención de documentos en formato .pdf

Siguiendo los pasos descritos en el punto anterior se obtiene un documento en formato `.dvi` (ver apartado 1.2.2). Esta es un formato ideal para trabajar y para imprimir, pero lo normal es obtener una copia de la versión final del trabajo en formato `.pdf` para su distribución. Para obtener esta copia tenemos dos opciones:

- Compilar utilizando `PDFLaTeX` en lugar de `LATEX`, para lo que habría que seleccionar `PDFLaTeX` en el menú rápido, o pulsar la tecla F6, o
- Transformar el archivo `.dvi` a `.pdf` mediante dos pasos, siendo ésta la **opción recomendada** en la mayoría de casos. El proceso consistiría en transformar primero a `.ps` y luego a `.pdf` mediante las acciones Herramientas → Dvi->PS (F4) y Herramientas → PS->PDF (F8).

Nótese que el proceso descrito de dos pasos es normalmente preferible al de un sólo paso consistente en la acción Herramientas → Dvi->PDF (F9), dado que esta última proporciona resultados insatisfactorios en múltiples situaciones. Nótese también que los compiladores `LATEX` y `PDFLaTeX` proporcionan resultados prácticamente idénticos (aunque en formatos diferentes). La mayor diferencia se encuentra en el tipo de imágenes que se pueden utilizar en cada caso, tal y como se comenta en el apartado 2.4.2, siendo la opción `LATEX` la preferida por este autor dado que permite utilizar imágenes de tipo vectorial (véase el apartado 1.3).

#### 2.2.5. Los errores en `LATEX`

Es perfectamente normal y prácticamente inevitable cometer errores a la hora de introducir comandos y estructuras en un documento `LATEX`. Lo importante es, por un lado, ser consciente de ello para tomarlo como algo natural que forma parte del proceso de creación del documento, y por otro lado, poder identificar y corregir dichos errores de manera sencilla.

En `TeXMaker`, todos los avisos o errores se muestran en la zona de mensajes, en la zona inferior de la ventana (ver figura 2.1). Si no existen errores, esta ventana mostrará el mensaje `Process exited normally`. En caso contrario, cuando existan errores se mostrará el mensaje `Process exited with error(s)`. En tal caso, los triángulo azules a la izquierda de la zona de mensajes permiten navegar por los distintos errores.

### 2.2.5.1. Ejemplos de errores en LATEX

Por ejemplo, si en el ejemplo anterior se escribe `\lateX` en lugar de `\LaTeX`, se obtendrá el mensaje

```
! Undefined control sequence.  
1.6 Este es mi primer documento \lateX
```

indicando que el comando `\lateX`, situado en la linea 6, es desconocido para LATEX y que debe ser corregido. Si en lugar de escribir `\end{document}` se introduce por error `\end{documen}`, el mensaje obtenido será

```
! LaTeX Error: \begin{document} ended by \end{documen}  
See the LaTeX manual or LaTeX Companion for explanation.  
Type H <return> for immediate help.  
...  
1.7 \end{documen}
```

indicando que en la línea 7, el entorno document no ha sido correctamente finalizado.

### 2.2.5.2. Propagación de errores

Si hace la prueba comentada en este último caso, verá que LATEX devuelve también dos errores adicionales que en realidad están causados por el primero de todos. En documentos grandes, la existencia de un error puede llevar a LATEX a dar multitud de avisos relacionados. Por ello, cuando se obtiene más de un aviso de error, es muy recomendable empezar por el primero y volver a compilar antes de continuar revisando los siguientes errores.

## 2.3. Estructura y contenido de un documento LATEX

Un documento LATEX está formado por:

- El texto que conformará el resultado final (en el ejemplo, la frase *Este es mi primer documento*), y
- Comandos (o palabras clave) precedidos del símbolo '\', como por ejemplo `\usepackage`, que permiten realizar multitud de tareas, tal y como veremos más adelante.

Pero además, un documento LATEX se divide en dos partes fundamentales: **préámbulo** y **cuerpo**, tal y como ilustra la tabla 2.2 para el documento de prueba anterior.

---

```
\documentclass[a4paper,12pt]{report}
\usepackage[utf8]{inputenc}           Preámbulo
\usepackage[spanish]{babel}
\begin{document}
Este es mi primer documento \LaTeX.    Cuerpo
\end{document}
```

---

Tabla 2.2: Estructura y contenido de un documento L<sup>A</sup>T<sub>E</sub>X

El **cuerpo** está formado por el contenido del documento (texto, figuras, tablas, ecuaciones, etc.) y su inicio y fin vienen definidos por las declaraciones `\begin{document}` y `\end{document}` respectivamente.

Pero antes de indicar cuál es el contenido del documento en sí, debemos dar a L<sup>A</sup>T<sub>E</sub>X indicaciones acerca de diversos aspectos generales tales como tipo de documento, tamaño de letra, tamaño de papel, idioma, etc. Toda esta información se da antes de la declaración `\begin{document}`, y constituye el **preámbulo** del documento.

### 2.3.1. *El preámbulo*

El documento, y por tanto el preámbulo del mismo, debe comenzar siempre con la orden

`\documentclass[opciones]{tipo de documento}`

que define el tipo de documento de entre las siguientes posibilidades:

- `article`
- `book`
- `report`
- `letter`

Las figuras 1.1 y 2.2 presentan ejemplos de documentos tipo `article` y `letter` respectivamente, mientras que este manual representa en sí un ejemplo de documento tipo `book`.

Por otro lado, se pueden definir diferentes opciones tales como:

- **Tamaño de letra:** 10pt, 11pt o 12pt.
- **Tamaño de página:** a4paper ( $29.7 \times 21$  cm), a5paper ( $21 \times 14.8$  cm), b5paper ( $25 \times 17.6$  cm), letterpaper ( $11 \times 8.5$  pulgadas), legalpaper ( $14 \times 8.5$  pulgadas) o executivepaper ( $10.5 \times 7.25$  pulgadas cm).
- **Orientación de página:** landscape para páginas apaisadas (la opción por defecto es la correspondiente a páginas verticales).
- **Formato de columna:** onecolumn o twocolumn.
- **Impresión a simple o doble cara:** oneside o twoside.

```
\documentclass[12pt,a4paper]{letter}
\usepackage[utf8x]{inputenc}
\usepackage[spanish]{babel}
\address{Universidad de Las Palmas de Gran Canaria \\ Campus Universitario de Tafira
\\ 35017 - Las Palmas de Gran Canaria}
\signature{El autor}
\begin{document}
\begin{letter}{Lectores Unidos \\ Carretera del libro, 4 \\ 35678 - Las Palmas de G.C.}
\opening{Estimados señores,}
```

Este es un ejemplo de un documento tipo `\texttt{letter}` preparada con `\LaTeX`.

```
\closing{Atentamente}
\end{letter}
\end{document}
```

Universidad de Las Palmas de Gran Canaria  
Campus Universitario de Tafira  
35017 - Las Palmas de Gran Canaria

7 de septiembre de 2010

Lectores Unidos  
Carretera del libro, 4  
35678 - Las Palmas de G.C.

Estimados señores,

Este es un ejemplo de un documento tipo `letter` preparado con `\LaTeX`.

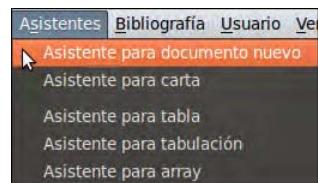
Atentamente

El autor

**Figura 2.2: Ejemplo de documento tipo carta. Arriba fichero tipo `\LaTeX`, formado por la cabecera y el cuerpo. Abajo, resultado tras ser procesado por `\LaTeX`**

El preámbulo es utilizado para dar más información general del documento. Por ejemplo, en la figura 1.1 vemos cómo se definen el título y autor del documento mediante los comandos `\title{}` y `\author{}` respectivamente, mientras en la figura 2.2 se da la dirección y firma del autor de la carta mediante los comandos `\address{}` y `\signature{}`. También se utiliza el preámbulo, entre otras cosas, para cargar paquetes con el objetivo de desempeñar diversas funciones, tal y como se comenta en la sección siguiente.

En *TeXMaker*, la creación de estos diferentes tipos de documento se facilita haciendo uso de la opción Asistentes → Asistente para documento nuevo, obteniendo un diálogo en el que podremos seleccionar algunas de las características más importantes del documento para que *TeXMaker* confeccione automáticamente el preámbulo del documento.



### 2.3.2. Los paquetes

Las posibilidades de un documento L<sup>A</sup>T<sub>E</sub>X pueden ser enormemente aumentadas mediante la introducción de paquetes, utilizando, para ello, el comando

```
\usepackage [Opciones] {Nombre del paquete}
```

En todos los ejemplos vistos hasta ahora se han incluido los paquetes

```
\usepackage [utf8] {inputenc}     y    \usepackage [spanish] {babel}
```

El primero de ellos permite el uso del teclado español y la introducción de caracteres acentuados como 'à' o 'ñ', mientras que el segundo establece el español, en lugar del inglés, como idioma del documento, siendo éstos sólo dos ejemplos, ya que existen innumerables paquetes con diversas posibilidades, como la ampliación de la cantidad de símbolos disponibles, la modificación del alfabeto utilizado, o la inclusión de colores y gráficos.

### 2.3.3. Los entornos

Una estructura ampliamente utilizada en L<sup>A</sup>T<sub>E</sub>X son los entornos. Su inicio y fin se indica generalmente mediante los comandos `\begin{}` y `\end{}`. El entorno más importante es el entorno `document`, que delimita el principio y fin del

contenido del documento. Otros ejemplos son los entornos `enumerate` o `center` que permiten la introducción de listas enumeradas y para centrar texto, tal y como muestran los ejemplos 2.1 y 2.2.

```
1 \documentclass[10pt]{article}      Los tipos de documento usuales en
2 \begin{document}                   LATEX son:
3 Los tipos de documento          1. article
4 usuales en \texttt{\textrm{\LaTeX}} son:   2. book
5 \begin{enumerate}                 3. report
6 \item article \item book        4. letter
7 \item report \item letter
8 \end{enumerate}
9 \end{document}
```

**Ejemplo 2.1: El entorno `enumerate`**

```
1 \documentclass[a4paper,10pt]{article}
2 \begin{document}
3 El siguiente texto aparece       El siguiente texto aparece centrado de-
4 centrado debajo:                bajo:
5 \begin{center}
6 Estoy                           Estoy
7                               centrado
8 centrado
9 \end{center}
10 \end{document}
```

**Ejemplo 2.2: El entorno `center`**

```
1 % Este texto esta comentado
2 % y no se procesa.           Este texto no esta comentado y se pro-
3 Este texto no esta comentado   cesa.
4 y se procesa. \\
5 Ahora se procesa. % Ahora no. Ahora se procesa.
6 \\ Y para dar el 100\%, se   Y para dar el 100%, se antepone la ba-
7 antepone la barra invertida.  rra invertida.
```

**Ejemplo 2.3: Comentando texto**

#### 2.3.4. *Los comentarios*

En ocasiones interesa evitar que LATEX procese un texto determinado. Es el caso por ejemplo de querer añadir comentarios aclaratorios al código o, por alguna razón, querer que LATEX ignore comandos, textos, entornos determinados. Esto se consigue anteponiendo el símbolo `%` al texto que no se desea procesar. El ejemplo 2.3 ilustra este uso.

### 2.3.5. Nuevos párrafos y líneas

El ejemplo 2.3 muestra también otros tres aspectos interesantes:

1. En la línea 3 del fichero .tex (izquierda), las palabras *siguiente* y *texto* han sido escritas separadas por múltiples espacios, mientras que el resultado (derecha) presenta la separación adecuada.
2. En el fichero .tex, la frase *El siguiente texto aparece centrado debajo:* se ha escrito en dos líneas, la 3 y la 4, con un retorno de carro (enter), y sin embargo aparece como una frase única en el documento final.
3. Las palabras *Estoy* y *Centrado* aparecen separadas por una línea en blanco, mientras en el resultado final, están una debajo de la otra.

¿Porqué ocurre esto? Entre otras cosas, L<sup>A</sup>T<sub>E</sub>X controla automáticamente el espacio que existe entre caracteres, palabras y líneas. Por esa razón, la existencia de uno o varios espacios entre palabras (punto 1) o incluso de un 'retorno de carro' (punto 2), significan exactamente lo mismo para L<sup>A</sup>T<sub>E</sub>X. Así, para comenzar un nuevo párrafo se necesitan dos o más retornos de carro consecutivos (punto 3).

### 2.3.6. Capítulos, secciones, subsecciones, ...

A la hora de redactar un documento, lo importante debe ser preocuparse por el contenido y por cómo se estructura éste. Por eso, la manera correcta de introducir capítulos, secciones, etc., en L<sup>A</sup>T<sub>E</sub>X es indicarle que se desea comenzar un nuevo capítulo o sección mediante los comandos correctos que, por orden jerárquico, son:

Capítulo	Sección	Subsección	Subsubsección
\chapter{}	\section{}	\subsection{}	\subsubsection{}

Tabla 2.3: Comandos básicos para estructurar un documento

De este modo, L<sup>A</sup>T<sub>E</sub>X se ocupará de la tipografía y de la numeración de las cabeceras de manera automática (véase ejemplo en la figura 2.3), además de ocuparse de la elaboración del índice.

```
\documentclass[a4paper,10pt]{article}
\title{On the Electrodynamics of Moving Bodies}
\author{Albert Einstein \date{1905}}
\begin{document} \maketitle
\section{Kinematics}
\subsection{Definition of Simultaneity}
Let us take a system [...]
\subsection{On the relativity of lengths and times}
The following reflexions [...]
\section{Electrodynamics}
\subsection{Transformation of the Maxwell-Hertz equations}
Let the Maxwell--Hertz eq. [...]
\subsection{Dynamics of the Slowly Accelerated Electron}
Let there be in motion [...]
\end{document}
```

# On the Electrodynamics of Moving Bodies

Albert Einstein

1905

## 1 Kinematics

### 1.1 Definition of Simultaneity

Let us take a system [...]

### 1.2 On the relativity of lengths and times

The following reflexions [...]

## 2 Electrodynamics

### 2.1 Transformation of the Maxwell-Hertz equations

Let the Maxwell--Hertz eq. [...]

### 2.2 Dynamics of the Slowly Accelerated Electron

Let there be in motion [...]

**Figura 2.3:** Ejemplo de uso de los comandos `section` y `subsection`

## 2.4. Para empezar a trabajar con LATEX

### 2.4.1. Modificando la tipografía: familias, perfiles y tamaños de letra

Existen diversas maneras de modificar la fuente utilizada: cambiar el tamaño, la familia, la forma, o el grosor, tal y como se ilustra en la tabla 2.4 donde, exceptuando el cambio de tamaño, las instrucciones de las dos primeras columnas son equivalentes, produciendo ambas el resultado mostrado en la tercera columna. Todos son comandos básicos de cualquier sistema LATEX a excepción de los comandos `\color{}` y `\textcolor[rgb]{0,0,1}{}`, que son proporcionados por el paquete `color`, por lo que será necesario introducir en el preámbulo el comando `\usepackage{color}`.

### 2.4.2. Incluyendo gráficos e ilustraciones

Existen dos maneras bien distintas de incluir gráficos e ilustraciones en LATEX:

- **Importar** imágenes realizadas previamente, tales como fotografías o dibujos guardados en nuestro disco duro, o
- **Componer** las ilustraciones con las órdenes específicas disponibles en LATEX para este fin.

Aunque es posible realizar ilustraciones de gran calidad utilizando directamente los comandos de LATEX, por lo general es mucho más sencillo y recomendable la primera de las opciones, es decir, realizar las gráficas e ilustraciones con programas externos (como puedan ser *gnuplot* para las gráficas o *InkScape* para las ilustraciones) e importarlos posteriormente.

A diferencia de los editores de textos más usuales, la importación de gráficos e ilustraciones en LATEX no se realiza 'cortando y pegando' la imagen, o insertándola en el texto, sino mediante el comando `\includegraphics`. El primer paso, sin embargo, es hacer uso del paquete `graphicx` mediante la inclusión de uno de los siguientes comandos en el preámbulo del documento:

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

Si se compila directamente con LATEX y las imágenes a introducir están en formatos .eps o .ps; o

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

Si se compila con PDFLATEX y las imágenes están disponibles en formato .jpg, .tif, .png o .pdf.

Cambio de tamaño		
{\tiny texto}		texto
{\scriptsize texto}		texto
{\footnotesize texto}		texto
{\small texto}		texto
{\normalsize texto}		texto
{\large texto}		texto
{\Large texto}		texto
{\LARGE texto}		texto
{\huge texto}	texto	texto
{\Huge texto}	texto	texto
Cambio de familia		
{\rmfamily texto}	\textrm{texto}	texto
{\ttfamily texto}	\texttt{texto}	texto
{\sffamily texto}	\textsf{texto}	texto
Cambio de forma		
{\upshape texto}	\upshape{texto}	texto
{\itshape texto}	\textit{texto}	<i>texto</i>
{\slshape texto}	\textsl{texto}	<i>texto</i>
{\scshape texto}	\textsc{texto}	TEXTO
Cambio de serie		
{\mdseries texto}	\textmd{texto}	texto
{\bfseries texto}	\textbf{texto}	<b>texto</b>
Enfatizar		
{\em texto}	\emph{texto}	<i>texto</i>
	\underline{texto}	<u>texto</u>
Cambio de color		
{\color{red} texto}	\textcolor{red}{texto}	texto
{\color{blue} texto}	\textcolor{blue}{texto}	texto

Tabla 2.4: Modificaciones al tipo de letra

Tal y como se explica en el apartado 1.3, los gráficos de tipo vectorial (.eps y .ps principalmente) son generalmente la mejor opción en documentos técnicos, tanto por calidad como por tamaño de ficheros, por lo que el autor recomienda la primera opción, es decir, \usepackage[dvips]{graphicx}.

Una vez cargado el paquete graphicx en el preámbulo, la inserción de una ilustración o gráfico se realiza mediante el comando

```
\includegraphics[Opciones]{Ruta a la imagen}
```

donde las opciones posibles se muestran en la tabla 2.5 y algunos ejemplos de uso en la figura 2.6.

Opción	Ejemplo	Descripción
keepaspectratio		Mantener la relación de aspecto
width	width=3.5cm	Determinar ancho
height	height=2.0cm	Determinar alto
scale	scale=0.5	Escalar el gráfico
angle	angle=45	Girar el gráfico

Tabla 2.5: Opciones del comando `includegraphics`

---

```
\includegraphics[keepaspectratio,height=.5cm]{tux.eps}
\includegraphics[width=1cm, height=0.5cm]{tux.eps}
\includegraphics[angle=270,scale=0.1]{tux.eps}
```

---



Tabla 2.6: Ejemplos de uso del comando `includegraphics`

#### 2.4.2.1. Posicionamiento y etiquetado automático de imágenes como objetos flotantes

Utilizando el comando `includegraphics` es posible introducir cualquier imagen en cualquier punto del documento, ya sea de manera aislada o en mitad de una línea. Por ejemplo, para insertar el símbolo podemos simplemente utilizar el comando `\includegraphics[height=10pt]{warning}` en medio del texto.

Sin embargo, el uso más común es la inserción de figuras, normalmente independientes, con numeración y pie de figura. Además, y aunque pueda parecer

trivial, el posicionamiento de las figuras (y tablas) en el documento puede ser problemático, y es deseable que se automatice al máximo.

Estas necesidades las afronta LATEX a través de lo que se denominan *objetos flotantes*, definidos por dos entornos, *figure* y *table*, para figuras (gráficos, ilustraciones, imágenes en general) y tablas respectivamente. En la compilación, LATEX busca la situación más óptima de todas las figuras y tablas para optimizar el espacio y la composición de todas las páginas. Por esta razón, el usuario no tiene, en principio, un control total sobre la posición final de los elementos, y debe habituarse a la idea de que LATEX buscará la mejor distribución (aunque, si fuera necesario, también es posible fijar la posición de un elemento flotante). En este caso, todas las figuras deben ser referenciadas en el texto a través del número de figura o tabla, y nunca a través de expresiones del tipo *la figura siguiente*.

Un entorno *figure* se define como:

```
\begin{figure}[indicador(es) de posición]
\includegraphics[]{} (u otro(s) comando(s))
\caption{pie de figura}
\label{etiqueta interna}
\end{figure}
```

donde los indicadores de posición son:

- t para colocar la figura preferentemente en la parte superior de la página.
- b para colocar la figura preferentemente en la parte inferior de la página.
- h para colocar la figura lo más cerca posible del lugar donde se define.
- p para que la figura se coloque en una página sólo de objetos flotantes (sin texto, pero si es posible con otras figuras o tablas).

Estos indicadores de posición pueden combinarse en cualquier número y orden. El caso extremo, `\begin{figure}[hptb]` indicaría a LATEX que puede colocar la figura tanto en la parte superior como inferior de una página con texto, o en una página exclusiva para figuras y tablas, en todo caso lo más cerca a esta posición posible. Por otro lado, la opción `\begin{figure}[h!]` solicita encarecidamente a LATEX que realice un esfuerzo extra por no mover la figura de lugar.

Nótese que dentro del entorno *figure* puede utilizarse, además del comando *includegraphics*, cualquier otra orden que se considere oportuna. Por ejemplo, una orden de gran utilidad es el comando *put*, que permite posicionar elementos adicionales dentro de la figura según el siguiente formato:

```
\put(coordenada x,coordenada y){elemento}
```

donde las coordenadas x e y definen la posición del elemento a introducir dentro del área de la figura, y el elemento en sí puede ser casi cualquier cosa: palabras, símbolos, ecuaciones, imágenes, ...

Así por ejemplo, la figura 2.1 es generada con el siguiente código:

```
\begin{figure}[h!]
\begin{center}
\includegraphics[width=0.80\textwidth,keepaspectratio]{TexMaker_Screenshot.eps}
\put(-380,185){\color{red} \tiny \underline{Menú principal}}
\put(-380,175){\color{red}\tiny \underline{Menús rápidos}}
\put(-100,140){\color{red}\tiny \underline{Zona de edición}}
\put(-100,30){\color{red}\tiny \underline{Zona de mensajes}}
\put(-310,100){\color{red}\tiny \underline{Zona de ayudas}}
\put(-310,90){\color{red}\tiny \underline{a la edición}}
\put(-103,177){\color{red} \circle{16}}
\put(-37,177){\color{red} \circle{16}}
\put(-245,29){\color{red} \circle{9}}
\put(-245,19){\color{red} \circle{9}}
\caption{Pantallazo de la interfaz de \TeX{} Maker} \label{TexMaker_Screenshot}
\end{center}
\end{figure}
```

donde podemos ver dos nuevos comandos: `\caption{}`, que permite definir el pie de la figura; y `\label{}`, con el que se definen etiquetas que permitirán posteriormente hacer referencia al objeto, tal y como se describe en el siguiente apartado.

#### 2.4.3. Etiquetando

La referencia a elementos numerados tales como figuras, tablas, apartados o ecuaciones es una constante en los documentos de carácter científico-técnico. Por otro lado, asignar la numeración de estos elementos, hacer las modificaciones pertinentes cuando hay cambios en el documento, y asegurarse de que las referencias están actualizadas, puede ser un trabajo muy engoroso si se realiza de manera manual. Por todo esto, el usuario nunca asigna numeraciones en L<sup>A</sup>T<sub>E</sub>X y, a la hora de hacer referencia, se utilizan etiquetas que, a la hora de compilar, el propio L<sup>A</sup>T<sub>E</sub>X se encarga de sustituir por el dato correcto.

Todo esto se consigue utilizando dos comandos. En primer lugar, el comando

```
\label{Etiqueta}
```

asigna una etiqueta a un elemento determinado, al que se hace referencia con el comando

```
\ref{Etiqueta}
```

donde la etiqueta puede estar compuesta por cualesquiera caracteres. Así, por ejemplo, a este apartado se le ha asignado la etiqueta `sec:etiquetas` escribiendo para comenzarlo el comando `\subsection{Etiquetando}` seguido de `\label{sec:etiquetas}`. Para hacer referencia a él más adelante, se utilizaría el comando `\ref{sec:etiquetas}`, cuyo resultado al compilar es: 2.4.3. Lo mismo ocurre con objetos flotantes como la figura 2.1, cuyo código se muestra en el apartado anterior y cuya etiqueta es `TexMaker_Screenshot`. El resultado de escribir `\ref{TexMaker_Screenshot}` es 2.1.

#### 2.4.4. Haciendo listas

Existen dos entornos fundamentales para generar listas: `enumerate` e `itemize`. Los ejemplos 2.4 y 2.5 ilustran el uso de estos entornos. Nótese que cada nueva entrada se identifica con el comando `\item`:

<sup>1</sup> Los entornos lista permiten:	
<sup>2</sup> <code>\begin{enumerate}</code>	Los entornos lista permiten:
<sup>3</sup> <code>\item clasificar</code>	1. clasificar
<sup>4</sup> <code>\item organizar</code>	2. organizar
<sup>5</sup> <code>\begin{enumerate}</code>	<sup>a)</sup> y anidar
<sup>6</sup> <code>\item y anidar</code>	<sup>b)</sup> unos dentro de otros
<sup>7</sup> <code>\item unos dentro de otros</code>	3. y muchas otras cosas
<sup>8</sup> <code>\end{enumerate}</code>	
<sup>9</sup> <code>\item y muchas otras cosas</code>	
<sup>10</sup> <code>\end{enumerate}</code>	

Ejemplo 2.4: El entorno `enumerate`

<sup>1</sup> Los entornos lista permiten:	
<sup>2</sup> <code>\begin{itemize}</code>	Los entornos lista permiten:
<sup>3</sup> <code>\item clasificar</code>	■ clasificar
<sup>4</sup> <code>\item organizar</code>	■ organizar
<sup>5</sup> <code>\begin{itemize}</code>	
<sup>6</sup> <code>\item y anidar</code>	• y anidar
<sup>7</sup> <code>\item unos dentro de otros</code>	• unos dentro de otros
<sup>8</sup> <code>\end{itemize}</code>	
<sup>9</sup> <code>\item y muchas otras cosas</code>	■ y muchas otras cosas
<sup>10</sup> <code>\end{itemize}</code>	

Ejemplo 2.5: El entorno `itemize`

#### 2.4.5. Incluyendo tablas

La estructura básica de una tabla se define mediante el entorno `tabular`, seguido de una indicación del número y alineación o tamaño de columnas, según el esquema

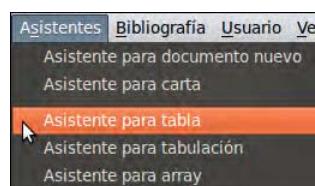
```
\begin{tabular}{Definición de columnas}
Texto fila 1-col 1 & ... & texto fila 1-última col\\
Texto fila 2-col 1 & ... & texto fila 2-última col\\
...
\end{tabular}
```

donde como definición de las columnas (nótese que no hay que indicar número ni formato de filas, sólo de columnas) se introducirá una de las letras l, r o c por cada columna, en función de que se desee que los elementos de la columna queden alineados a la izquierda, la derecha o centrados, respectivamente. En estos casos, el ancho de la columna se ajustará automáticamente al elemento más ancho de la misma, que no podrá tener más de una línea. Sin embargo, para determinar un ancho fijo y, además, tener la posibilidad de que el texto ocupe varias líneas, se debe utilizar la letra `p{ancho}`. Por otro lado, los símbolos y comandos de mayor interés en este entorno se muestran en la tabla 2.7.

Símbolo o comando	Descripción
&	nueva columna
\\" ó \newline	nueva fila
	línea vertical entre columnas (a especificar en la definición de columnas)
\hline	línea horizontal entre filas

Tabla 2.7: Símbolos y comandos de interés en el entorno `tabular`

Todos estos conceptos se ven más claros en los ejemplos 2.6 a 2.8, que muestran la población de Boston en función del año y diferentes posibilidades en cuanto a líneas divisorias y formato de columnas. Por otro lado, *TeXMaker* proporciona ayuda a la hora de elaborar tablas mediante el diálogo Asistentes → Asistente para tablas.



```

1 \begin{center}
2 \begin{tabular}{lr}
3 \textbf{Year} &
4 \textbf{Population}\\
5 1800 & 24,937 \\
6 1900 & 560,892 \\
7 2000 & 589,141
8 \end{tabular}
9 \end{center}

```

Year	Population
1800	24,937
1900	560,892
2000	589,141

Ejemplo 2.6: El entorno tabular sin divisiones

```

1 \begin{center}
2 \begin{tabular}{l|r}
3 \textbf{Year} &
4 \textbf{Population}\\
5 \hline 1800 & 24,937 \\
6 \hline 1900 & 560,892 \\
7 \hline 2000 & 589,141
8 \end{tabular}
9 \end{center}

```

Year	Population
1800	24,937
1900	560,892
2000	589,141

Ejemplo 2.7: El entorno tabular con divisiones

```

1 \begin{center}
2 \begin{tabular}{|p{1.5cm}|c|}
3 \hline \centering \textbf{Year} &
4 \textbf{Population}\\
5 \hline \centering 1800 & 24,937 \\
6 \hline \centering 1900 & 560,892 \\
7 \hline \centering 2000 & 589,141 \\
8 \hline
9 \end{tabular}
10 \end{center}

```

Year	Population
1800	24,937
1900	560,892
2000	589,141

Ejemplo 2.8: El entorno tabular con divisiones y una columna de ancho definido

#### 2.4.5.1. Tablas como objetos flotantes

El entorno tabular puede ser utilizado prácticamente en cualquier situación, y es muy útil, en general, para organizar elementos en cuadrículas. En el caso de que se desee insertar una tabla clásica, la mejor opción es hacerlo como un objeto flotante y con su pie de tabla y numeración correspondiente. Para ello, introduciremos el código que define la tabla en un entorno table:

```
\begin{table}[indicador(es) de posición]
\centering
\begin{tabular}{def. de columnas}
.....
\end{tabular}
\caption{pie de tabla}
\label{etiqueta interna}
\end{table}
```

donde los indicadores de posición son exactamente los mismos que se definieron en el apartado 2.4.2.1. A modo de ejemplo, la tabla 2.5 es generada mediante el siguiente código:

```
\begin{table}[h]
\centering
\begin{center}
\begin{tabular}{lll}
\textbf{Opción} & \textbf{Ejemplo} & \textbf{Descripción} \\ \hline
\texttt{keepaspectratio} & & Mantener la relación de aspecto \\
\texttt{width} & \texttt{width=3.5cm} & Determinar ancho \\
\texttt{height} & \texttt{height=2.0cm} & Determinar alto \\
\texttt{scale} & \texttt{scale=0.5} & Escalar el gráfico \\
\texttt{angle} & \texttt{angle=45} & Girar el gráfico \\ \hline
\end{tabular}
\caption{Opciones del comando \texttt{includegraphics}}
\label{OpcionesIncludegraphics}
\end{center}
\end{table}
```

## 2.4.6. Incluyendo símbolos y fórmulas matemáticas

Además de por su universalidad y por la excelente calidad compositiva de sus textos, L<sup>A</sup>T<sub>E</sub>X es especialmente conocido y apreciado por sus amplias posibilidades a la hora de escribir ecuaciones matemáticas y símbolos. Veremos, además, que la introducción de elementos matemáticos en los textos que hemos ido desarrollando hasta ahora no es una tarea complicada, sino más bien todo lo contrario.

### 2.4.6.1. El modo matemático y sus entornos

Hasta este momento, hemos estado utilizando L<sup>A</sup>T<sub>E</sub>X en lo que podríamos denominar *modo texto*, que es el *estado por defecto* de L<sup>A</sup>T<sub>E</sub>X. Existen algunos comandos relacionados con la inserción de símbolos que son accesibles desde este *modo texto*. Por lo general, sin embargo, para introducir fórmulas o símbolos matemáticos aprovechando al máximo las posibilidades de L<sup>A</sup>T<sub>E</sub>X se-

rá necesario hacerlo en el denominado *modo matemático* según alguna de las siguientes variantes:

- **Modo matemático en línea**, utilizado para insertar símbolos y pequeñas fórmulas siguiendo la línea normal del texto, como por ejemplo  $E = mc^2$ . El **modo matemático en línea** se puede comenzar y terminar mediante el símbolo  $\$^2$ :

```
$ Fórmula $
```

- **Modo matemático destacado**, utilizado para presentar fórmulas separadas del texto, normalmente centradas y numeradas, como al escribir

$$E = mc^2 \quad (2.1)$$

El **modo matemático destacado**, con ecuaciones numeradas, se obtiene con el entorno `equation`:

```
\begin{equation}
Fórmula
\end{equation}
```

mientras los entornos `equation*` o `displaymath` producen ecuaciones destacadas sin numerar<sup>3</sup>.

#### 2.4.6.2. Algunos conceptos importantes

Al igual que ocurre con la composición del texto, LATEX tiende a gestionar de manera autónoma la composición de los elementos dentro de una fórmula siguiendo los estándares establecidos en la edición de textos matemáticos. Dichos estándares establecen, por ejemplo, que las constantes se escriban en tipología *Roman* y las variables en cursiva, y también definen los espacios entre variables, constantes y operadores. De todo esto se ocupa, por lo general, LATEX, de modo que el usuario sólo tiene que intervenir para hacer un *ajuste*

<sup>2</sup> También se pueden usar los símbolos `\(` y `\)` o el entorno `math`, pero el símbolo `$` es más recomendable.

<sup>3</sup> Este es el entorno básico, pero existen también otros entornos que facilitan funcionalidades extra como la agrupación de múltiples ecuaciones, la escritura de ecuaciones que ocupan varias líneas, la alineación de ecuaciones en puntos concretos, etc.

fino en contadas ocasiones. Eso sí, cuando lo tiene que hacer, tiene a su disposición una inmensa capacidad de maniobra para obtener el resultado deseado. Por esta razón, los espacios en blanco en modo matemático, por ejemplo, no tienen ningún efecto, de modo que las expresiones  $E=mc^2$  y  $E = m c^2$  producen ambas exactamente el mismo resultado:  $E = mc^2$ .

#### 2.4.6.3. Los símbolos en LATEX

La cantidad de símbolos disponible en LATEX es enorme. Sin embargo, un aspecto a tener muy en cuenta es que la gran mayoría de esos símbolos no están disponibles de partida, sino que estarán a disposición del usuario tras cargar el paquete adecuado.

Por ejemplo, un proveedor muy importante de símbolos (y también de entornos) matemáticos es la *American Mathematical Society* (AMS), y es altamente recomendable cargar siempre sus paquetes incluyendo las siguientes órdenes en el preámbulo del documento:

```
\usepackage{amsmath}  
\usepackage{amsfonts}  
\usepackage{amssymb}
```

Una gran mayoría de símbolos vienen dados en forma de comandos. Por ejemplo, la letra griega  $\lambda$  viene dada por el comando `\lambda`, y el símbolo  $\sum$  por el comando `\sum`.

TEXMaker ofrece en su lateral izquierdo una relación de los símbolos más usuales divididos en 5 grupos: Símbolos de relación , flechas , otros símbolos , delimitadores  y letras griegas , que aparecen en la zona de ayudas a la edición (ver figura 2.1). Al hacer clic sobre el símbolo deseado, TEXMaker introducirá el comando adecuado.

Una relación completa de todos los símbolos disponibles, puede encontrarse en el documento "The Comprehensive LATEX Symbol List", disponible en [www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf](http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf). El documento contiene símbolos tanto para el modo texto como para el modo matemático. Al consultar este documento es importante notar que gran parte de los símbolos son provistos por paquetes específicos que hay que cargar en el preámbulo del documento. Por ejemplo, la tabla 233 de dicho documento muestra varios símbolos relacionados con la seguridad, y para obtener los símbolos de modo texto  (\textsf{Biohazard}) o  (\textsf{Radioactivity}) debe cargarse el paquete `marvosym` que indica el nombre de la tabla.

#### 2.4.6.4. Modos de texto dentro del modo matemático

Por defecto, LATEX compone todas las letras contenidas en modo matemático considerando que se trata de variables. Sin embargo, en muchas ocasiones es necesario introducir algo de texto en una ecuación. El comando básico para introducir texto en modo matemático es el comando `\text{}`. El ejemplo 2.9 muestra la comparación entre los resultados obtenidos utilizando o no este comando, viéndose que la opción correcta es la mostrada en la línea 3.

```

1 \begin{center}
2 $oferta=f(demanda)$ \\
3 $\text{oferta}=f(\text{demanda})$           oferta =  $f(demanda)$ 
4 \end{center}

```

#### Ejemplo 2.9: Introducción de texto en modo matemático

El comando `\text{}` compone el texto utilizando la fuente y opciones generales del documento. Sin embargo, otros comandos permiten la elección de una fuente determinada, tal y como se muestra en la tabla 2.8.

Comando	Descripción	Resultado
<code>\mathbf{Texto}</code>	Negrilla	Texto
<code>\mathit{Texto}</code>	Cursiva	<i>Texto</i>
<code>\mathrm{Texto}</code>	Roman	Texto
<code>\mathsf{Texto}</code>	Sans Serif	Texto
<code>\mathtt{Texto}</code>	Typewriter	Texto
<code>\mathcal{R,C,I}</code>	Caligráfico	$\mathcal{R,C,I}$
<code>\mathfrak{R,C,I}</code>	Euler Fraktur	$\mathfrak{R,C,I}$
<code>\mathbb{R,C,I}</code>	Blackboard	$\mathbb{R,C,I}$

Tabla 2.8: Familias de texto modo matemático

#### 2.4.6.5. Algunos elementos importantes en modo matemático

La tabla 2.9 muestra algunos de los elementos más utilizados en modo matemático, acompañados de algunos ejemplos para facilitar su comprensión.

Exponentes e índices: $a^{\{superíndice\}}$ , $a_{\{subíndice\}}$		
$a^b$	$a_b$	$a_{b_j}^{c^2}$
$a^b$	$a_b$	$a_{\{b_j^k\}}^{c^2}$
Fracciones: $\frac{\text{Numerador}}{\text{Denominador}}$		
$\frac{a}{b}$	$\frac{a^b}{c_d}$	$\frac{a+b}{g+h}+2$
$\frac{a}{b}$	$\frac{a^b}{c_d}$	$\frac{a+b}{g+h}+2$
Raíces: $\sqrt[n]{\text{radicando}}$		
$\sqrt{a}$	$\sqrt[b]{a}$	$\sqrt[b+c]{a_i}$
$\sqrt{a}$	$\sqrt[b]{a}$	$\sqrt[b+c]{a_i}$
Sumatorios: $\sum_{\text{límite inferior}}^{\text{límite superior}}$		
$\sum_j a_j$	$\sum_{i=1}^n a_i$	$\sum_{i=1}^n \sum_{j=1}^m a_{ij}$
$\sum_j a_j$	$\sum_{i=1}^n a_i$	$\sum_{i=1}^n \sum_{j=1}^m a_{ij}$
Integrales: $\int_{\text{límite inferior}}^{\text{límite superior}}$		
$\int x dx$	$\int_a^b x dx$	$\int_a^b f(x) dx$
$\int x dx$	$\int_a^b x dx$	$\int_a^b f(x) dx$

Tabla 2.9: Algunos elementos importantes en modo matemático

Comando	Ejemplo	Resultado
<code>normal</code>	$x \, dx$	$xdx$
<code>\,</code>	$x \, , dx$	$x dx$
<code>\:</code>	$x \, : dx$	$x dx$
<code>\;</code>	$x \, ; dx$	$x dx$
<code>\quad</code>	$x \quad dx$	$x \quad dx$
<code>\quad</code>	$x \quad dx$	$x \quad dx$
<code>\negthinspace</code>	$x \negthinspace dx$	$xdx$

Tabla 2.10: Los espacios en modo matemático

#### 2.4.6.6. Espacios y puntos en modo matemático

La tabla 2.10 muestra la forma de controlar de manera precisa el espaciado entre caracteres, mientras la tabla 2.11 muestra los distintos grupos de puntos disponibles en el modo matemático. El ejemplo 2.10 ilustra también el uso de estos grupos de puntos.

Comando	Ejemplo	Resultado
\ldots	a_1 \ldots a_2	$a_1 \dots a_2$
\cdots	a_1 \cdots a_2	$a_1 \cdots a_2$
\vdots	a_1 \vdots a_2	$a_1 \vdots a_2$
\ddots	a_1 \ddots a_2	$a_1 \ddots a_2$

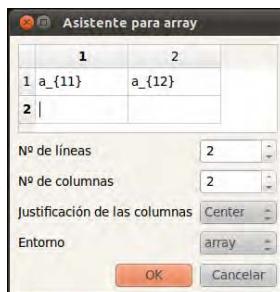
Tabla 2.11: Los puntos en modo matemático

#### 2.4.6.7. Matrices y vectores

Los vectores y las matrices en modo matemático son creados, generalmente, utilizando el entorno array

```
\begin{array}{Definición de columnas}
Texto fila 1-col 1 & ... & texto fila 1-última col\\
Texto fila 2-col 1 & ... & texto fila 2-última col\\
...
\end{array}
```

donde la definición de las columnas se realiza de manera análoga al caso del entorno tabular descrito en la sección 2.4.5. El entorno array funciona exclusivamente en modo matemático, por lo que generará un error si se intenta utilizar en modo texto. De nuevo, *TeXMaker* nos pueden facilitar la tarea de generar este entorno mediante sus asistentes, tal y como se muestra en la figura 2.4. El ejemplo 2.10 ilustra el uso de este comando muy útil y versátil.

Figura 2.4: Asistente de *TeXMaker* para el entorno array

```

1 \begin{center}
2 \begin{equation*}
3 \begin{array}{cccc}
4 a_{11} & a_{12} & \cdots & a_{1n} \\
5 \vdots & \vdots & \ddots & \vdots \\
6 a_{n1} & a_{n2} & \cdots & a_{nn}
7 \end{array}
8 \end{equation*}
9 \end{center}

```

**Ejemplo 2.10: El entorno array****2.4.6.8. Paréntesis de tamaño variable**

Otro elemento muy necesario a la hora de construir expresiones matemáticas es el paréntesis de tamaño variable. Este tipo de paréntesis se adapta automáticamente en tamaño al contenido que abarque, y se añade mediante los comandos `\left` y `\right` seguidos del tipo de paréntesis a utilizar, siendo los más comunes los paréntesis comunes ( `y` ), los corchetes [ `y` ], y la barra vertical | . Las llaves { `y` }, también de uso común, deben ser precedidas de la barra invertida, debiendo introducirse `\{` y `\}` respectivamente. Así, cerrar una expresión (que puede ser tan compleja como sea necesario), con corchetes de tamaño variable, consiste en comenzar la expresión con el comando `\left[` y terminarla con `\right]`. Los ejemplos 2.11 a 2.13 muestran el uso de este elemento.

```

1 \begin{center}
2 \begin{equation*} \left[ \begin{array}{cccc}
3 a_{11} & a_{12} & \cdots & a_{1n} \\
4 \vdots & \vdots & \ddots & \vdots \\
5 a_{n1} & a_{n2} & \cdots & a_{nn} \end{array} \right]
6 \end{array} \right]
7 \end{array} \right]
8 \end{equation*}
9 \end{center}

```

**Ejemplo 2.11: El entorno array con corchetes**

Un caso interesante (véase ejemplo 2.13) es aquel en el que un paréntesis debe abrirse pero no cerrarse. En este caso, y aunque el paréntesis no deba cerrarse, es necesario indicar a LATEX dónde termina el ámbito de ese paréntesis, de tal forma que pueda calcular su tamaño. Esto se indica mediante el comando `\right .` colocado al final de la expresión, tal y como se ilustra en el ejemplo 2.13.

```

1 \begin{center}
2 \begin{equation*}
3 x=\frac{\left| \begin{array}{cc}
4 b_1 & a_{12} \\
5 b_2 & a_{22} \end{array} \right|}{\left| \begin{array}{cc}
6 a_{11} & a_{12} \\
7 a_{21} & a_{22} \end{array} \right|}
8 \end{array} \right| \end{array} \right|
9 \end{equation*}
10 \end{center}

```

$$x = \frac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}$$

### Ejemplo 2.12: El uso de la barra vertical

```

1 \begin{center}
2 \begin{equation*}
3 f(x) = \left\{ \begin{array}{ll}
4 \begin{array}{c} \text{ si } x < a \\ -1 \end{array} \\
5 \phantom{f(x)=} \text{ si } x = a \\ 
6 \begin{array}{c} \text{ si } x > a \\ +1 \end{array} \\
7 \end{array} \right. .
8 \end{array} \right. \\
9 \end{equation*}
10 \end{center}

```

$$f(x) = \begin{cases} -1 & \text{si } x < a \\ 0 & \text{si } x = a \\ +1 & \text{si } x > a \end{cases}$$

### Ejemplo 2.13: El uso de las llaves

```

1 \begin{equation}\label{ec:l}
2 I=\int_a^b f(x)\,\mathrm{d}x
3 \end{equation}
4 Eq. \ref{ec:l} shows
5 a definite integral.
6
7 Eq. \eqref{ec:l} shows
8 a definite integral.

```

$$I = \int_a^b f(x) \, dx \quad (2.2)$$

Eq. 2.2 shows a definite integral.

Eq. (2.2) shows a definite integral.

### Ejemplo 2.14: Referenciando ecuaciones

#### 2.4.6.9. Referencias a ecuaciones

Al igual que ocurre con figuras o tablas, es posible asignar etiquetas a las ecuaciones para poder hacer referencia a ellas en cualquier punto del documento. Para asignar una etiqueta, basta con utilizar el comando `\label{}` en el interior de un entorno matemático numerado, como por ejemplo el entorno

equation. Posteriormente, el uso del comando `\ref{}` proporciona la numeración de dicha ecuación. Por otro lado, el paquete `amsmath` proporciona también el comando `\eqref{}`, que devuelve el número de la ecuación entre paréntesis (véase ejemplo 2.14).

#### 2.4.7. Escribiendo documentos grandes

A la hora de escribir un documento de cierto tamaño, como pueda ser un libro, una tesis doctoral o un informe de múltiples capítulos, es muy interesante poder trabajar cada parte del documento en un fichero diferente. De este modo se evita tener todo el trabajo en un fichero muy grande, multiplicando así la posibilidad de cometer errores y la dificultad de subsanarlos, dificultando la navegación por el documento, y haciendo que LATEX tenga que procesar cada vez la totalidad del documento, con el consiguiente consumo de tiempo. En este caso existen dos opciones, los comandos `\input{}` e `\include{}`.

##### 2.4.7.1. El comando `input`

Si LATEX encuentra, por ejemplo, el comando `\input{Parte2.tex}`, sustituirá dicha línea por el contenido del fichero Parte2.tex, y procesará el documento como tal. De este modo puede introducirse contenido en cualquier parte del documento, incluso en el preámbulo. Este comando suele utilizarse para partes concretas de un documento: una demostración matemática compleja, una parte del preámbulo diseñada por uno mismo y que se utiliza con asiduidad, distintas secciones de un artículo o un report, o un texto que es utilizado en distintos documentos y que es más sencillo de mantener en un fichero único.

En este caso, la única manera de evitar que LATEX procese un `input` determinado es comentándolo (anteponiendo el símbolo %) o borrándolo, lo que en muchos casos no es una buena solución. Por esto, y sobre todo para libros, tesis, etc., se utiliza preferentemente el comando `\include{}`.

##### 2.4.7.2. El comando `include`

El comando `\include{}` suele utilizarse para separar los capítulos de un documento en distintos archivos independientes. Para ello, se construye un documento *maestro* que hace referencia a los demás, como ocurriría en el siguiente caso:

```
\documentclass[a4paper,11pt,twoside,spanish]{book}
\usepackage[utf8x]{inputenc}
\usepackage[spanish]{babel}

\begin{document}
```

```
\tableofcontents % genera el índice de manera automática
\include{c1_intro}
\include{c2_latex}
\include{c3_gnuplot}

\end{document}
```

donde los tres capítulos se introducen utilizando este comando, lo que contribuye a trabajar de forma más ordenada y segura. El comando `\include{}` tiene las siguientes características:

- No puede utilizarse en el preámbulo, sino únicamente en el cuerpo del documento.
- Genera una nueva página antes y después de la llamada al comando `\include{}`, por lo que es ideal para ser utilizado por capítulos.
- No puede utilizarse de forma recursiva. Es decir, un archivo llamado desde un comando `\include{}` no puede, a su vez, contener más comandos `\include{}`.
- Puede usarse junto al comando `\includeonly{lista de archivos a procesar}`. Este comando, que debe colocarse en el preámbulo del documento, indica de entre todos los `\include{}` que existan en el documento, cuáles serán los únicos que se procesen (ver ejemplo más adelante).
- Genera y mantiene los archivos auxiliares de todos los `\include{}` por separado, de forma que incluso si posteriormente se utiliza el comando `\includeonly{}`, L<sup>A</sup>T<sub>E</sub>X conservará la información necesaria sobre el paginado, el índice y las referencias de las partes no procesadas.

El siguiente código ilustra el uso del comando `\includeonly{}`. A diferencia del ejemplo anterior, en el que se procesarían los tres capítulos, en este caso se procesaría únicamente la introducción (el primer capítulo), sin necesidad de modificar nada más en el documento, gracias a la inclusión del comando `\includeonly{c1_intro}`.

```
\documentclass[a4paper,11pt,twoside,spanish]{book}
\usepackage[utf8x]{inputenc}
\usepackage[spanish]{babel}
\includeonly{c1_intro}

\begin{document}

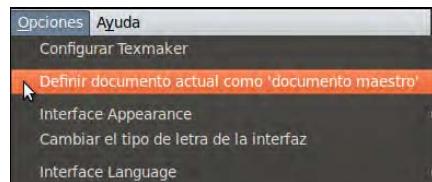
\tableofcontents % genera el índice de manera automática
\include{c1_intro}
\include{c2_latex}
\include{c3_gnuplot}

\end{document}
```

#### 2.4.7.3. A tener en cuenta a la hora de compilar

Al trabajar de esta manera, será necesario tener abiertos varios ficheros a la vez: el documento maestro (como el mostrado en el ejemplo anterior) y todos los ficheros adicionales incluidos mediante los comandos `input` o `include`. Si, por ejemplo, se está trabajando en uno de los capítulos de un libro (que no contiene preámbulo, dado que éste se define en el documento maestro), se obtendría un error si se tratara de compilarlo directamente. Esto es así porque el fichero que hay que compilar en realidad es el maestro, que contiene el preámbulo y todos los comandos necesarios para *llamar* a las distintas partes del documento.

Sin embargo, tampoco es cómodo ni productivo tener que cambiar al documento maestro cada vez que se quiera compilar el documento. Para solucionar esto, lo único que se debe hacer es indicar a *TeXMaker* cuál es el documento maestro, de tal forma que cuando se trate de compilar cualquiera de los ficheros asociados, *TeXMaker* sepa exactamente lo que debe hacer. Esto se consigue situándose en el documento maestro y seleccionando la acción `Opciones` → `Definir documento actual como 'documento maestro'`.



## 2.5. Y mucho más ...

Siendo esta manual tan solo una introducción, son incontables los comandos, entornos y posibilidades de  $\text{\LaTeX}$  que se han quedado en el tintero. En el capítulo 3 se tratará también la gestión de la bibliografía con  $\text{\LaTeX}$ , pero aún así, lo mostrado aquí es solamente un aperitivo de la potencia de este sistema. Para cada necesidad existe una solución en  $\text{\LaTeX}$ , generalmente de la mano de un nuevo paquete. Los hay para química, para música, para agrupar filas y columnas en una tabla, y para muchas otras aplicaciones particulares. Por poner un ejemplo ilustrativo, un paquete muy útil en general es `beamer`, que permite generar presentaciones en formato `.pdf` de muy alta calidad, y que también permite la generación de pósters en diversos formatos.

A partir de lo visto en este manual, y a medida que comience a utilizar  $\text{\LaTeX}$ , usted empezará a preguntarse cómo realizar tareas concretas o cómo solucionar los problemas o dudas que seguro le van a surgir. Además de toda la ayuda que puede encontrarse en internet, se recomienda especialmente la consulta de algún libro de referencia como pueden ser el de Helmut Kopka y Patrick W. Daly [5] y el de Bernardo Cascales *et al.* [6].

## CAPÍTULO 3

### Gestión de la bibliografía en $\text{\LaTeX}$ : *BiBTeX + JabRef*

### 3.1. Introducción

Vimos en el capítulo 2 que L<sup>A</sup>T<sub>E</sub>X automatiza y facilita la tarea de numerar y referenciar elementos tales como fórmulas, tablas y figuras mediante la utilización de etiquetas y referencias. Un método muy similar se puede utilizar para la generación de las citas bibliográficas de un documento, de modo que tras asignar una etiqueta identificativa a cada entrada bibliográfica, sea L<sup>A</sup>T<sub>E</sub>X quien se encargue de realizar de manera autónoma la tarea de numeración de las entradas bibliográficas y de sus referencias a lo largo del texto.

Existen dos maneras de realizar esta tarea. La primera de ellas consiste en definir, en algún lugar del documento, la lista de las entradas bibliográficas en un entorno específico denominado `thebibliography`, siendo el usuario quien deberá asumir la responsabilidad de ordenar la lista y de dar formato a cada una de las entradas, tal y como se describe en la sección 3.2. La segunda opción es almacenar, en un archivo `.bib` independiente del documento, la información correspondiente a todas las entradas bibliográficas, e indicar a L<sup>A</sup>T<sub>E</sub>X que busque en dicho archivo la información que necesite. Esta segunda opción es generalmente más aconsejable, sobre todo en el caso de documentos relativamente grandes, tal y como veremos en la sección 3.3, donde para facilitar la tarea, se mostrará el uso del programa *JabRef*.

### 3.2. La bibliografía a mano: El entorno `thebibliography`

El entorno `thebibliography` es una de las dos posibilidades existentes a la hora de generar la bibliografía de un documento. Generalmente se sitúa al final del documento, antes de la instrucción `\end{document}`, y dentro de él se listan, precedidos del comando `\bibitem`, todas las entradas bibliográficas, según el formato siguiente:

```
\begin{thebibliography}{leyenda de muestra}
...
\bibitem[Leyenda]{Etiqueta} Texto
...
\end{thebibliography}
```

```

1 \begin{thebibliography}{1}
2
3 \bibitem{Parmelee:1967}
4 Parmelee RA. Building–Foundation
5 Interaction Effects.
6 J Eng Mech Div ASCE 1967;
7 93(EM2):131–152.
8
9 \bibitem{Wolf:1985} Wolf JP.
10 Dynamic soil–structure interaction.
11 Prentice–Hall, Englewood Cliffs,
12 NJ, 1985.
13
14 \bibitem{Clough} Clough RW,
15 Penzien J. Dynamics of structures.
16 McGraw–Hill, 1993.
17
18 \bibitem[Av]{Aviles:2002}
19 Avil \'e;s J, Su\'arez M.
20 Effective periods and dampings
21 of building–foundation systems
22 including seismic wave effects.
23 Eng Struct 2002;24:553–562.
24
25 \end{thebibliography}

```

# Bibliografía

- [1] Parmelee RA. Building-Foundation Interaction Effects. J Eng Mech Div ASCE 1967; 93(EM2):131-152.
- [2] Wolf JP. Dynamic soil-structure interaction. Prentice-Hall, Englewood Cliffs, NJ, 1985.
- [3] Clough RW, Penzien J. Dynamics of structures. McGraw-Hill, 1993.
- [Av] Avilés J, Suárez M. Effective periods and dampings of building-foundation systems including seismic wave effects. Eng Struct 2002;24:553-562.

### Ejemplo 3.1: El entorno thebibliography

El ejemplo 3.1 muestra una aplicación del mismo que puede utilizarse para ilustrar cada uno de los elementos del entorno. Puede verse que cada entrada va precedida del comando `\bibitem` seguido de la etiqueta que la identificará. Para citar una entrada determinada se utiliza el comando `\cite{etiqueta}`. Por ejemplo, el comando `\cite{Clough}` generará como resultado [3]. La numeración de las sucesivas entradas es generada de forma automática por LATEX a no ser que se establezca una leyenda de antemano, como en la última entrada del ejemplo, donde se establece la leyenda `Av` mediante la instrucción `\bibitem[Av]{Aviles:2002}`. Así, el comando `\cite{Aviles:2002}` genera como resultado: [Av]. Sin embargo, el uso más común es el mostrado en las tres primeras entradas, donde se da a LATEX la libertad de generar las leyendas automáticamente.

El texto escrito a continuación del comando `\bibitem[leyenda]{etiqueta}` es reproducido de manera literal a la hora de generar la bibliografía, y es el usuario el que debe estar atento a su formato. Nótese que en el ejemplo, las

tildes son introducidas utilizando el comando de L<sup>A</sup>T<sub>E</sub>X para ello (`\'{letra}`). En este caso, la escritura directa de una vocal acentuada (u otros caracteres especiales) daría lugar a error a la hora de compilar.

Por último, al entorno `thebibliography` hay que pasarle una leyenda de muestra. Llamamos aquí leyenda a los caracteres que aparecen entre corchetes a la izquierda de cada entrada bibliográfica y que sirven para identificarla. En el ejemplo 3.2 las leyendas son 1, 2, 3 y Av. El único cometido de la leyenda de muestra es mostrar a L<sup>A</sup>T<sub>E</sub>X cuál va a ser la longitud de la leyenda más larga, de modo que pueda hacer las sangrías necesarias. El ejemplo se ha escrito a propósito de manera errónea para ilustrarlo. Así, la última entrada tiene una leyenda de dos caracteres que ha forzado al texto (el que comienza por *Avilés J*) hacia la derecha, desalineándolo del resto de entradas. Si la leyenda de prueba hubiera sido '77' o 'Av' en lugar de '1', todas las entradas se hubiesen desplazado ligeramente hacia la derecha de modo que quedaran todas alineadas.

También es importante notar que, cada vez que se modifique la bibliografía, es necesario ejecutar L<sup>A</sup>T<sub>E</sub>X dos veces para que toda la información y todas las referencias sean correctamente generadas.

### 3.3. Automatizar la bibliografía

El uso del entorno `thebibliography` tiene tres inconvenientes:

1. El usuario es el responsable de que el formato de todas y cada una de las entradas bibliográficas sea acorde al formato exigido o deseado según el destino del documento y, por supuesto, es también el responsable de que todas las referencias guarden un formato uniforme. En caso de querer cambiar algún aspecto del formato, el usuario deberá corregir manualmente todas y cada una de las entradas.
2. El usuario es responsable de ordenar las entradas ya sea por orden alfabético o por orden de aparición en el texto.
3. Para reutilizar una entrada en un documento diferente, debe primero localizarse ésta, copiarse al nuevo documento, y posteriormente modificar su formato si fuera necesario.

Sin embargo, existe otra posibilidad para resolver el problema de la gestión de las referencias bibliográficas. Se trata de generar una especie de *registro o base de datos* con todas las referencias que puedan ser de utilidad y al que L<sup>A</sup>T<sub>E</sub>X (a través de un programa asociado llamado *BiBTeX*) pueda acudir para buscar única y exclusivamente aquellas entradas que necesite. Además, en este registro se especifican por separado cada uno de los elementos que pueden

componer una entrada (nombre y apellido del autor o autores, título, año, editorial, ...) en función de su naturaleza. Es decir, para un libro se introducirá información relativa a, entre otros, la editorial y ciudad de publicación, mientras para un artículo científico será relevante el nombre, volumen y páginas de la revista de publicación.

Estos *registros o bases de datos* son tan solo archivos de texto plano, con extensión .bib, y con un formato determinado. Podrían escribirse o modificarse manualmente, pero es mucho más sencillo utilizar una de las múltiples herramientas disponibles para este fin. *JabRef* es una muy buena opción.

### 3.3.1. Obtención e instalación de JabRef

*JabRef* es un gestor de referencias bibliográficas que genera los archivos .bib utilizados por  $\text{\LaTeX}$ . Se trata de un programa de software libre disponible para  $\Delta$  GNU/Linux (directamente desde los repositorios de la mayor parte de distribuciones), para  $\text{\textcircled{A}}$  Mac OS X y para  $\text{\textcircled{W}}$  MS Windows.

En cualquier caso, el programa puede descargarse libremente desde su página web <http://jabref.sourceforge.net/> haciendo clic en el botón *Download latest stable version*. Tras una sencilla instalación, al ejecutarlo por primera vez obtendremos simplemente la ventana mostrada en la figura 3.1.

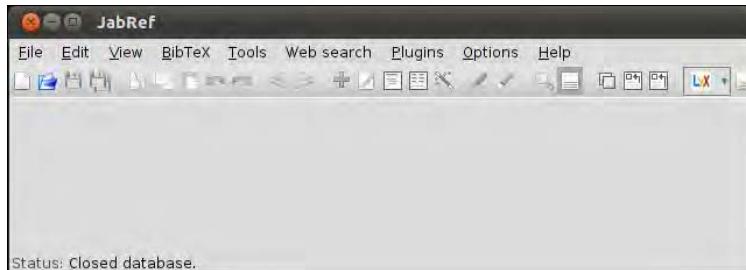


Figura 3.1: Interfaz inicial de JabRef

### 3.3.2. Creación de archivo .bib con JabRef

Primero debe crearse una nueva base datos haciendo clic en el botón nuevo  $\square$ . Ahora ya puede generarse una nueva entrada bibliográfica haciendo clic en el botón 'nueva entrada'  $\text{\textcolor{blue}{+}}$ . Al hacerlo, surgirá el cuadro de diálogo mostrado en la figura 3.2 que permite seleccionar el tipo de entrada de entre numerosas posibilidades. Al seleccionar una de ellas, *JabRef* solicita únicamente los campos relevantes a la selección. La figura 3.3 muestra la ventana del programa durante el proceso de creación del archivo .bib para este manual.



Figura 3.2: Cuadro de diálogo para la selección de nueva entrada

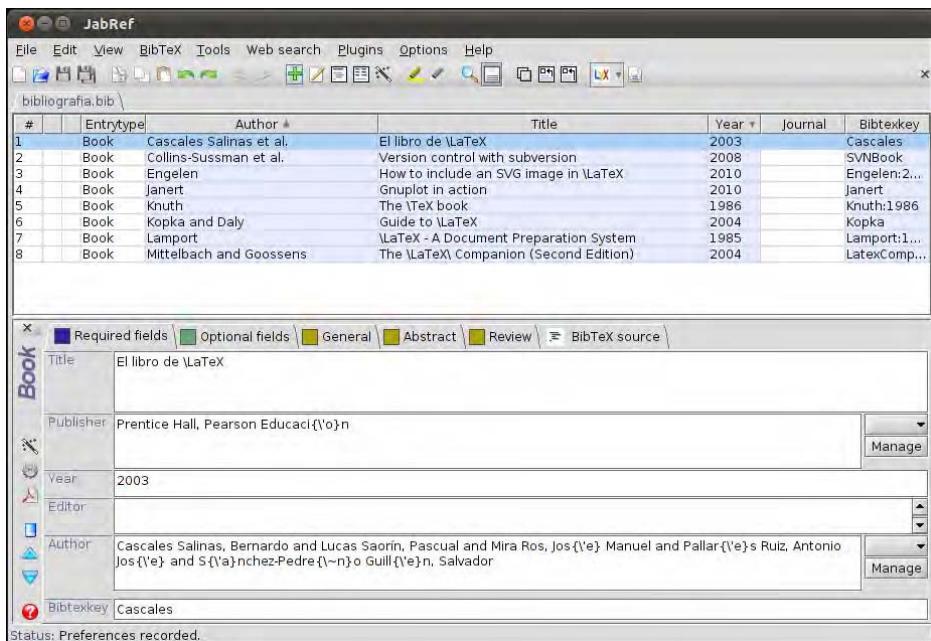


Figura 3.3: Introducción de elementos en la base de datos

Un campo de vital importancia, y común a todos los tipos de documentos, es el campo Bibtexkey. Aquí debe indicarse la etiqueta del documento, que servirá posteriormente para citarlo utilizando, como se vio en el apartado anterior, el comando `\cite{etiqueta}`. El texto de la etiqueta es de libre elección, pero debe intentarse que sea lo más informativo posible para facilitar la elaboración posterior del documento.

La base de datos debe ser guardada en un archivo .bib, y posteriormente podrá ser abierta y modificada en cualquier momento utilizando *JabRef*, otras aplicaciones similares, o incluso de forma manual.

### 3.3.2.1. *Sobre el formato del campo 'autores' y otros aspectos a tener en cuenta*

Un aspecto a tener en cuenta, y que se ilustra en el ejemplo de la figura 3.3, es la necesidad de evitar caracteres especiales en los campos de la base de datos. En su lugar, deben utilizarse los comandos *LATEX* para tal efecto, siendo los más comunes, `\`a` para la tilde aguda sobre un carácter, `"a` para la diéresis y `\~n` para la ñ.

Otro aspecto importante es el formato del campo 'autores'. Las principales reglas a tener en cuenta son las siguientes:

- En función del estilo seleccionado para el documento, será *LATEX* el encargado de dar formato a los autores. El formato podrá indicar que se especifique nombre y apellido completos (Andrew Murray), inicial punto apellido (A. Murray), apellido coma nombre (Murray, Andrew), etc. En todo caso, y para que *LATEX* pueda realizar este trabajo, la entrada bibliográfica debe contener, en la medida de lo posible, toda la información posible en el formato indicado en los puntos siguientes.
- El nombre puede indicarse de la forma *Nombres Apellido* (válido únicamente para un único apellido) o de la forma *Apellido(s)*, *Nombre(s)*. Esta última forma es la que debe utilizarse siempre que el autor tenga más de un apellido, dado que en el caso de la primera forma, sólo la última palabra será considerada apellido. Respecto al nombre, por contra, pueden indicarse varios nombres en cualquiera de las formas.
- En el caso de que existan varios autores, todos deben separarse entre sí utilizando la palabra *and*, tal y como se ilustra en el ejemplo de la figura 3.3.

### 3.3.3. *Generación de la bibliografía con BiBTEX*

Finalmente, una vez generado el archivo (o los archivos) .bib con todas las entradas bibliográficas necesarias, sólo queda introducirlas en el documento y hacer que sea procesada por *LATEX* realizando los siguientes pasos:

1. **Citar los documentos** a incluir en la bibliografía utilizando el comando `\cite{etiqueta}`. Únicamente serán introducidos en la bibliografía

<b>x</b>	<b>Estilo de leyenda</b>	<b>Ejemplo</b>	<b>Ordenación</b>
plain	Número entre corchetes	[1]	Alfabética
unsrt	Número entre corchetes	[1]	Por orden de cita
alpha	Nombre del autor y año	[Cas03]	Alfabética
abbrv	Nombre del autor y año	[Cas03]	Alfabética

**Tabla 3.1: Estilos de bibliografía por defecto**

aquellas entradas bibliográficas citadas en el texto<sup>4</sup>. (Obviamente, cada cita se realiza en el lugar del texto correspondiente.)

2. **Especificar el estilo de bibliografía.** El estilo determina la manera en que se formatean las referencias. Aunque hay muchos otros disponibles desde diversas fuentes, los estilos incorporados por defecto en la distribución LATEX son: plain, unsrt, alpha y abbrv. El estilo se especifica con el comando `\bibliographystyle{estilo}`. (ver tabla 3.1 para más información).
3. **Especificar el archivo BiBTEX a utilizar**, con el comando `\bibliography{ nombre de archivo BiBTeX }` en el lugar donde debe aparecer la bibliografía. El nombre del archivo se introduce sin incluir la extensión .bib.
4. **Ejecutar LATEX, luego BiBTEX y finalmente LATEX dos veces.** Este paso hay que seguirlo siempre que se modifique algún elemento relacionado con la bibliografía (cuando se añadan o eliminan referencias, cuando se modifique el archivo .bib, ...), y es necesario porque es BiBTEX quien construye la bibliografía a partir de la información generada por LATEX en su primera ejecución, quien a su vez incorpora la bibliografía y las referencias en sus siguientes dos ejecuciones.

Siguiendo estos pasos, se habrá generado ya la bibliografía del documento. Nótese que para ejecutar BiBTEX sólo es necesario elegir la opción **BibTeX** en el menú rápido (y pulsar la flecha azul) o pulsar la tecla F11.

---

<sup>4</sup> Existe también el comando `\nocite{referencia}`, que permite incluir en la bibliografía documentos no citados en el texto.

CAPÍTULO 4  
**Representación gráfica de datos y funciones**  
**con *gnuplot***

## 4.1. ¿Qué es *gnuplot* y para qué sirve?

*Gnuplot* es un programa multiplataforma para la generación de representaciones gráficas 2D y 3D de datos y funciones, escrito inicialmente por Thomas Williams y Colin Kelley, y desarrollado posteriormente por muchos otros colaboradores. Este programa es capaz generar gráficos bidimensionales sencillos como el de la figura 4.1, gráficos bidimensionales compuestos como el de las figuras 4.2 o 4.3, o gráficos tridimensionales como el de la figura 4.4.

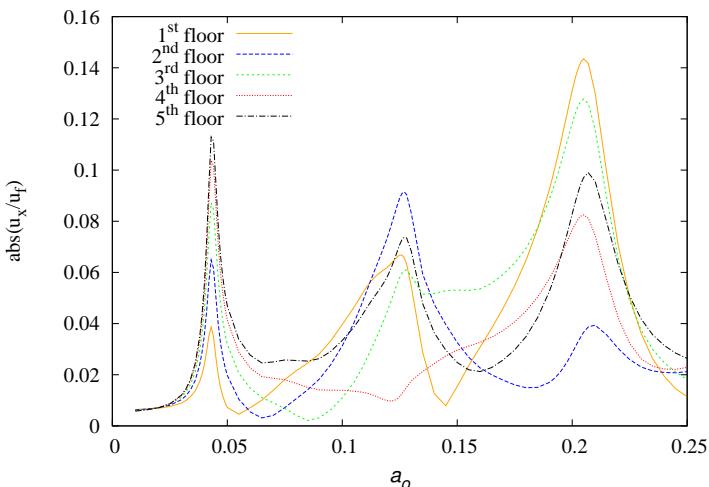


Figura 4.1: Ejemplo de figura sencilla

Estas representaciones pueden obtenerse en una ventana interactiva o pueden guardarse en distintos tipos de ficheros de salida, como .eps, .jpeg o .png. Por otro lado, la interacción con *gnuplot* tiene lugar a través de la línea de comando o a través de un fichero que contiene todas las instrucciones. Así, la figura 4.5 muestra lo que el usuario podrá encontrarse al iniciar *gnuplot* por primera vez. No se trata de un programa que se maneje a golpe de ratón. A primera vista, esta característica puede generar rechazo entre los usuarios no acostumbrados a manejar programas no basados en 'ventanitas'. Sin embargo, esta misma característica le confiere un gran potencial y, una vez aprendidos los conceptos básicos, se podrá comprobar la productividad que ofrece este planteamiento. Por esa razón, este capítulo está pensado para iniciar a todo usuario en el uso y los conceptos básicos del programa, para que a partir de ahí, cualquier duda concreta pueda ser satisfecha con la propia ayuda del programa o con su magnífica documentación (que puede obtenerse fácilmente tal y como se describe en la próxima sección).

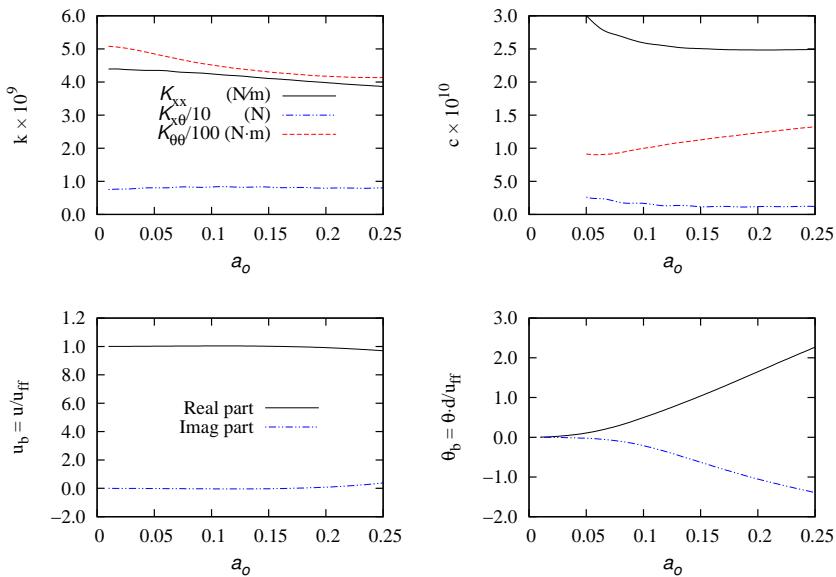


Figura 4.2: Ejemplo de figura compuesta

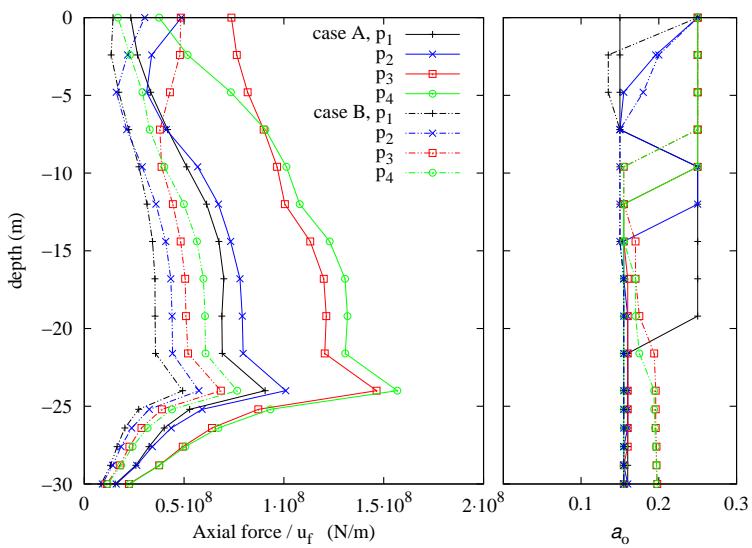


Figura 4.3: Ejemplo de figura compuesta

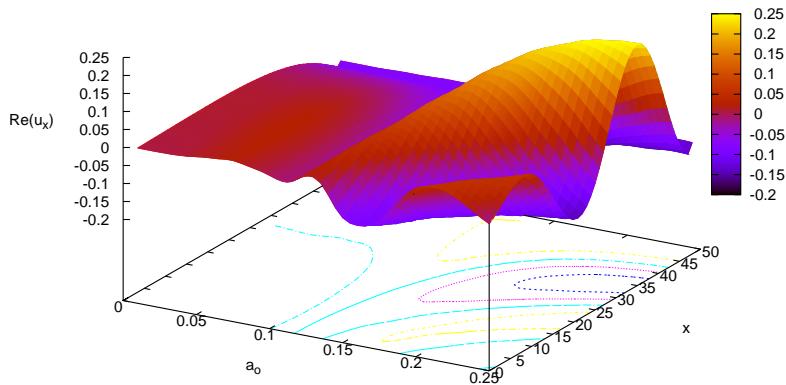


Figura 4.4: Ejemplo de representación tridimensional

```
lpadron@mmc09: ~
File Edit View Search Terminal Help
$ gnuplot
      G N U P L O T
      Version 4.4 patchlevel 2
      last modified Wed Sep 22 12:10:34 PDT 2010
      System: Linux 2.6.38-11-generic
      Copyright (C) 1986-1993, 1998, 2004, 2007-2010
      Thomas Williams, Colin Kelley and many others
      gnuplot home:    http://www.gnuplot.info
      faq, bugs, etc:  type "help seeking-assistance"
      immediate help: type "help"
      plot window:    hit 'h'
Terminal type set to 'wxt'
gnuplot>
```

Figura 4.5: Inicio del programa gnuplot

## 4.2. Obtención, instalación y ejecución de gnuplot

<http://www.gnuplot.info> es la página oficial del programa. Las secciones más útiles para comenzar son, por un lado, 'FAQ' y 'Documentation', donde puede obtenerse la documentación completa que describe todas las posibilidades del programa, y 'Download' donde puede encontrarse un enlace (que lleva a <http://sourceforge.net/projects/gnuplot/files/>) desde el que descargar la última versión del programa. Éste es el lugar más indicado para descargar la versión para Windows del programa. Sin embargo, en el caso de GNU/Linux, *gnuplot* está por lo general disponible directamente en los repositorios de la mayoría de las distribuciones (para instalarlo en Ubuntu o en Debian, puede utilizarse 'Synaptic' o puede teclearse en una terminal `sudo apt-get install gnuplot`. Por último, 'fink' es la manera más directa de instalar este programa en Mac OS X.

Para ejecutar el programa en GNU/Linux o en Mac OS X, basta con teclear *gnuplot* en una terminal. En Windows, al descomprimir el fichero .zip descargado de la página oficial, aparece una carpeta *gnuplot*, dentro de la que se puede encontrar otra carpeta llamada *binary*, donde podremos ejecutar el archivo *gnuplot.exe*.

La página oficial ofrece también los términos exactos de su licencia. En resumen, *gnuplot* es un programa gratuito, que puede ser descargado, instalado, copiado y distribuido libremente, pero que al mismo tiempo tiene derechos de autor y una licencia que impone ciertas restricciones sobre la modificación y redistribución del código fuente.

Los comandos básicos de *gnuplot* para la generación de representaciones gráficas son *plot* y *splot*, que permiten obtener representaciones 2D y 3D respectivamente. Con estos comandos pueden representarse tanto funciones analíticas como datos numéricos obtenidos de ficheros ordenados por columnas, tal y como se verá en las siguientes secciones.

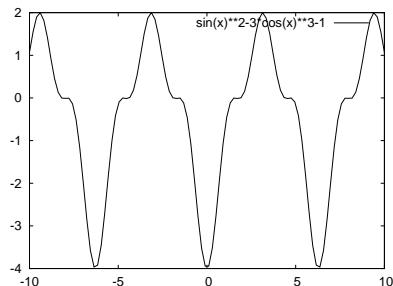
## 4.3. Representación de funciones analíticas

El ejemplo 1 muestra cómo obtener la representación de una función, en este caso  $f(x) = \sin^2(x) - 3\cos^3(x) - 1$ . El texto a introducir para ello es '`plot sin(x)**2-3*cos(x)**3-1`' ('gnuplot>' representa el *prompt* del programa, a la espera de las instrucciones del usuario).

Si es necesario operar con una misma función en varias ocasiones, o si la función a representar es muy compleja y se desea dividirla en partes, *gnuplot* permite definir funciones con las que operar posteriormente, tal y como se muestra en el ejemplo 2. En este ejemplo se ve también que es posible representar tantas funciones como sea necesario, tan solo separándolas por comas.

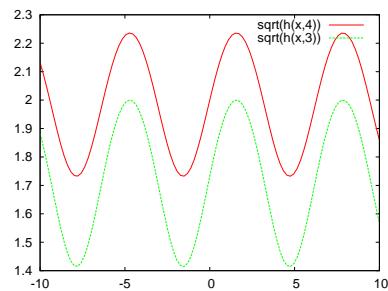
### Ejemplo de gnuplot 1

```
gnuplot> plot sin(x)**2-3*cos(x)**3-1
```



### Ejemplo de gnuplot 2

```
gnuplot> h(x,y)=sin(x)+y
gnuplot> plot sqrt(h(x,4)), sqrt(h(x,3))
```



En ambos ejemplos se utilizan funciones ya implementadas en *gnuplot*. En el primer caso se utilizan las funciones trigonométricas y en el segundo la función raíz cuadrada (*sqrt*). Un resumen de las funciones disponibles se presenta en la sección 4.13.

## 4.4. Representación de ficheros de datos

Además de funciones analíticas, *gnuplot* permite representar datos almacenados en ficheros de texto ordenados por columnas, como por ejemplo el mostrado en la figura 4.6, que servirá para ilustrar el funcionamiento de *gnuplot* en este caso.

Las primeras dos filas son líneas de comentarios que *gnuplot* ignorará por completo, y que se comienzan mediante el símbolo *#*. El fichero está compuesto de tres columnas de datos numéricicos. En este ejemplo concreto las dos últimas columnas están expresadas en formato científico (p.e. '0.10000801· 10<sup>1</sup>). En general, los datos pueden ser introducidos como enteros ('3' o '-31'), como

## fichero de texto ejemplo.txt

```
# Lineas de comentarios
# ao          Re(u)           Im(u)
0.010  0.10000801E+01 -0.26834899E-04
0.050  0.10005918E+01 -0.12152502E-02
0.100  0.99943820E+00 -0.19848360E-02
0.150  0.99843828E+00 -0.65669224E-03
0.200  0.99565479E+00  0.24174479E-02
0.250  0.99020154E+00  0.98897752E-02
0.300  0.98284652E+00  0.21703846E-01
0.350  0.97076299E+00  0.35900216E-01
0.400  0.94890684E+00  0.53588111E-01
0.450  0.91634947E+00  0.78585762E-01
0.500  0.87644722E+00  0.10885392E+00
0.550  0.82933838E+00  0.13923218E+00
0.600  0.77643901E+00  0.16247139E+00
0.650  0.71150718E+00  0.17132179E+00
0.700  0.63568312E+00  0.17189025E+00
0.750  0.56022760E+00  0.15787099E+00
0.800  0.47931664E+00  0.12669280E+00
0.850  0.40085335E+00  0.93869947E-01
0.900  0.34218439E+00  0.58409746E-01
0.950  0.30255307E+00  0.14728833E-01
1.000  0.28013064E+00 -0.32391664E-01
```

Figura 4.6: fichero de ejemplo 'ejemplo.txt'

reales ('3.0', '-3.1e1' o '55.8e-4') o como imaginarios ('{4,1}' o '{-2.0,6.0}'). La distinción entre número real y número entero es importante en relación a las operaciones aritméticas, principalmente en cuanto a la división, ya que de la operación '1/5' produce el valor '0', mientras que de '1.0/5.0' se obtiene el valor '0.2'.

El ejemplo 3 muestra cómo es posible representar datos del fichero 'ejemplo.txt' guardado previamente en el mismo directorio desde el que se está ejecutando *gnuplot*. Pueden distinguirse tres nuevas características:

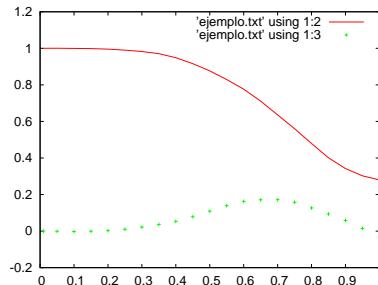
- Para representar los datos presentes en un archivo, se escribe su nombre entre comillas simples después del comando *plot*.
- Para especificar qué columna se desea representar, se utiliza la opción *using* seguida de la columna a representar en el eje horizontal (abscisas) y de la columna con los datos a representar en el eje vertical (ordenadas), ambos separados por dos puntos. En este caso, se ha representado la columna 2 frente a la 1, y la columna 3 frente a la 1.

- Para especificar que se desea utilizar líneas en lugar de los puntos de datos aislados, se utiliza la opción `with lines`.

### Ejemplo de gnuplot 3

---

```
gnuplot> plot 'ejemplo.txt' using 1:2
 $\downarrow$  with lines, 'ejemplo.txt' using 1:3
```



El símbolo  `$\downarrow$`  indica que la línea es continuación de la anterior, y aparece partida en la figura únicamente debido a falta de espacio.

## 4.5. Manipulación y transformación de los datos contenidos en ficheros

Puede ser muy interesante realizar alguna operación sobre los datos presentes en el fichero antes de representarlos, del tipo de operaciones sencillas por filas que se podrían realizar en una hoja de cálculo: operaciones trigonométricas, operaciones algebraicas entre distintas columnas, raíces cuadradas, o combinación de varias de ellas. El ejemplo 4 ilustra algunas posibilidades.

### Ejemplo de gnuplot 4

---

plot 'ejemplo.txt' using 1:(sqrt(\$2))	Representa las raíces cuadradas de los datos de la 2 <sup>a</sup> columna
plot 'ejemplo.txt' using 1:(((\$2+\$3)/100)	Representa la suma de la 2 <sup>a</sup> y 3 <sup>a</sup> columnas dividido entre 100
plot 'ejemplo.txt' using 1:(\$2**(\$3))	Representa la 2 <sup>a</sup> columna elevada al dato de la 3 <sup>a</sup> columna
plot 'ejemplo.txt' using 1:(cos(\$3)**2)	Representa el coseno cuadrado de la 3 <sup>a</sup> columna

El símbolo `$` indica la columna de donde extraer los datos para operar. Además de los operadores usuales para las operaciones de suma (+), resta (-), multiplicación (\*), división (/) y potencia (\*\*), *gnuplot* dispone de muchos más, que pueden ser consultados en el propio programa utilizando el comando `help`

operators. En todo caso, la sección 4.14 presenta un resumen de los mismos. Además, también pueden utilizarse las funciones implementadas de *gnuplot*, resumidas en la sección 4.13.

## 4.6. Exportación de gráficos: elección de formato de imagen y de fichero de salida

La verdadera utilidad de *gnuplot* reside en aspectos tales como la generación de gráficos en el formato de imagen más adecuado a su uso posterior, la automatización en la generación de distintos gráficos de igual formato, o la posibilidad de reutilizar el trabajo de edición de gráficos. Todo esto requiere ser capaz de exportar los gráficos a diversos formatos de imagen (png, jpeg, ps, etc.), y que esto pueda hacerse sin necesidad de introducir los comandos uno a uno cada vez que surge la necesidad de generar un nuevo gráfico. Como se verá a continuación, la primera necesidad se cubre con la elección de un formato y de un fichero de salida para la imagen generada, mientras que la segunda necesidad queda satisfecha por la posibilidad de escribir los comandos en un fichero de texto plano (que llamaremos fichero de procedimiento) que *gnuplot* podrá interpretar posteriormente.

La generación y exportación de imágenes a un formato de nuestra elección (y distinto de la pantalla de trabajo, que es la salida seleccionada por defecto en *gnuplot*) se realiza en dos pasos fundamentales:

- Elección del formato de imagen a través del comando:

```
set term <nombre de terminal>
```

- Elección del archivo de salida a través del comando:

```
set output <'nombre de archivo'>
```

*Gnuplot* ofrece un gran número de posibles *terminales*, es decir, de formatos de salida. La lista completa puede ser obtenida con el comando `set term`. Entre las muchas posibilidades pueden encontrarse los formatos png, jpeg o postscript entre muchos otros.

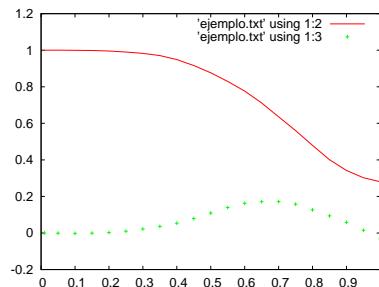
Tal y como se comenta en la introducción, los formatos de tipo vectorial representan múltiples ventajas. En el caso que nos ocupa, donde es necesario generar representaciones gráficas de datos y/o funciones, los formatos de tipo vectorial permiten reproducir imágenes de la máxima calidad y resolución con un tamaño de archivo muy pequeño, además de permitir ediciones posteriores, lo que hace que formatos como el postscript (ps) o el encapsulated postscript (eps) constituyan la elección más adecuada. Además, y como se vio en el capítulo correspondiente, estos formatos de imagen pueden ser directamente importados en L<sup>A</sup>T<sub>E</sub>X. Por todas estas razones, se va

a estudiar específicamente la generación de archivos *postscript*, si bien el proceso para generar las imágenes en cualquier otro formato es totalmente análogo y no presenta, por lo general, ninguna dificultad añadida.

### Ejemplo de gnuplot 5

---

```
gnuplot> set term postscript enhanced
      color "Helvetica" 22
gnuplot> set output 'MiPrimerPs.ps'
gnuplot> plot 'ejemplo.txt' using 1:2
      with lines, 'ejemplo.txt' using 1:3
gnuplot> set output
```



El ejemplo 5 muestra los comandos básicos para generar imágenes *postscript*. El primer comando establece el terminal del tipo *postscript* con texto ampliado (opción `enhanced`, cuya utilidad se verá en la sección 4.8), en color, con fuente 'Helvetica' de tamaño de letra 22 pt. (Para consultar los detalles y opciones de cada terminal, basta teclear `help set term <nombre de terminal>`, por ejemplo, `help set term postscript`).

El segundo comando establece que todo lo generado a partir de este momento (y hasta que se establezca un *output* distinto, se cambie de terminal, o se salga del programa con el comando *exit*) será volcado en el archivo de nombre *MiPrimerPs.ps*, de nueva creación. En este caso, el tercer comando (un *plot*) imprime el contenido de la figura.

Por último, el comando *set output* tiene como objetivo asegurarse de que *gnuplot* escribe en el fichero toda la información necesaria. Es algo parecido a cerrar el fichero, y aunque no siempre es necesario, es muy recomendable para asegurarse de que no se tendrán problemas por esta causa.

## 4.7. Aprovechando el trabajo: los ficheros de procedimiento

Teclear comandos uno a uno directamente sobre la terminal es muy útil para la visualización rápida de resultados. Sin embargo, para la generación y exportación de gráficos, es mucho más productivo escribir los comandos en un fichero de texto y hacer, posteriormente, que *gnuplot* interprete dicho fichero. Esto permitirá hacer pequeñas modificaciones posteriores sin mucho esfuerzo, producir multitud de gráficos similares con el mismo fichero de procedimiento (o varios similares), etc.

Para ilustrar esta posibilidad, retomemos el ejemplo 5. La gráfica puede ser generada introduciendo en *gnuplot* los tres comandos indicados arriba uno por uno, o pueden escribirse primero en un fichero de texto plano (utilizando cualquier herramienta tal como el 'bloc de notas' , el 'gedit' o el 'TextEdit' ). De este modo podríamos escribir un fichero de texto similar al siguiente:

fichero de texto generar-grafico.gnuplot

```
set term postscript enhanced color "Helvetica" 22
set output 'MiPrimerPs.ps'
plot 'ejemplo.txt' using 1:2 with lines, 'ejemplo.txt' using 1:3
set output
```

El nombre de archivo y su extensión son completamente libres. En este caso, el nombre que se le ha dado al archivo es `generar-grafico.gnuplot`, pero se podría haber optado por cualquier otro, como por ejemplo `MiFigura.g`. Una vez generado este archivo de procedimiento, existen dos opciones para hacer que *gnuplot* la ejecute, dependiendo de si *gnuplot* está o no en ejecución. Ambas opciones se ilustran en la figura 4.7: ejecutar en una terminal *gnuplot <nombre de archivo>* (arriba) o utilizar el comando `load '<nombre de archivo>'` desde el propio *gnuplot* (abajo). Ambas formas producen resultados idénticos, si bien la primera opción es la más recomendable por no necesitar entrar en *gnuplot* y por evitar la posibilidad de que alguna variable modificada en la ejecución del programa pueda influir inadvertidamente en otro resultado.

La gran mayoría de los ejemplos presentados a partir de este punto tendrán el formato de un fichero de procedimiento.

#### 4.8. Títulos, etiquetas, leyendas y rangos

Ahora que ya conocemos las posibilidades básicas de *gnuplot*, sólo nos falta introducirnos en la manera de dar formato al gráfico según las necesidades del usuario. El ejemplo 6 ilustra cómo introducir títulos, etiquetas, leyendas y rangos para un gráfico.

La función de las dos primeras líneas es ya conocida de los apartados anteriores. Los comandos `set xlabel` y `set ylabel` configuran las etiquetas de los ejes *x* e *y* respectivamente. Se ve aquí la utilidad de la opción `enhanced` del comando `set term`, gracias a la cual pueden introducirse subíndices y superíndices haciendo uso del formato de *LATEX*. Como se verá en la sección 4.11, esta opción da también la posibilidad de introducir símbolos diversos en los textos del gráfico.

La siguiente línea (`set title`) establece el título del gráfico, no necesario por lo general cuando la figura va a ser insertada en un documento, dado que irá

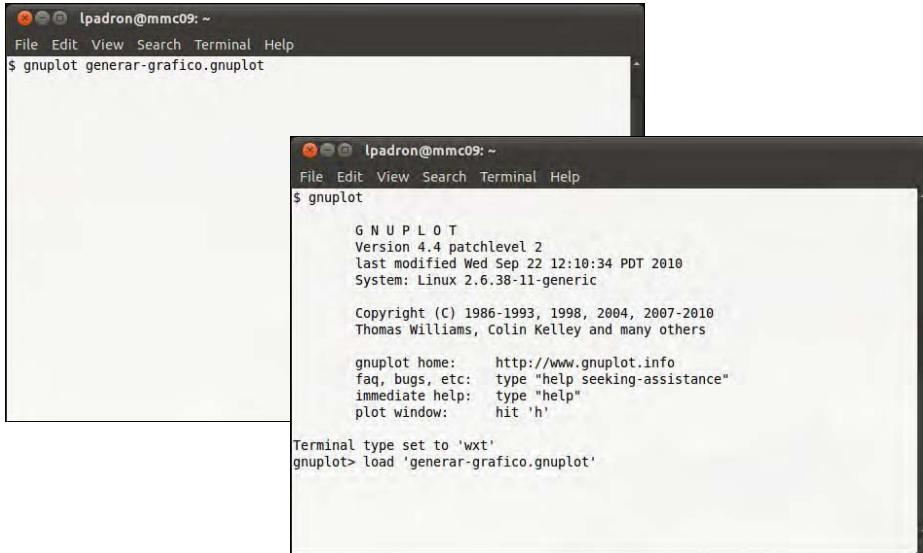


Figura 4.7: Dos maneras de ejecutar un fichero de procedimientos

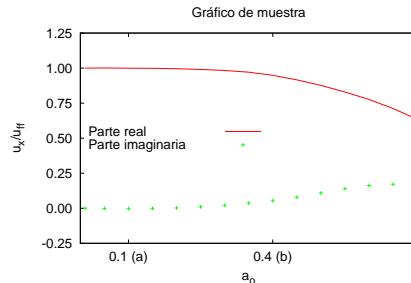
acompañado por un pie de figura. Para más detalles sobre cómo obtener la tilde de la palabra *Gráfico* (línea 5 del ejemplo) véase la sección 4.11.

### Ejemplo de gnuplot 6

```

set term postscript enhanced color
` "Helvetica" 22
set output 'TituloYMas.ps'
set xlabel "a_0"
set ylabel "u_x/u_{ff}"
set title "Gr-a{\302}fico de muestra"
set key left center
set key Left
set xrange [0:0.7]
set yrange [-0.25:1.25]
set ytics autofreq 0.25
set xtics autofreq ("0.1 (a)" 0.1,
` "0.4 (b)" 0.4)
plot 'ejemplo.txt' using 1:2 with lines
` title 'Parte real', 'ejemplo.txt' using 1:3
` title 'Parte imaginaria'

```



Las dos líneas siguientes establecen propiedades de la leyenda. Tales propiedades son establecidas mediante las opciones del comando `set key`. Las múltiples opciones disponibles pueden ser consultadas con el comando `help key`. Algunas de las más interesantes son la opción `off`, que oculta la leyenda; el conjun-

to de opciones `left|right|center` y `top|bottom|center` que establecen su posición horizontal y verticalmente; las opciones `Left|Right` que establecen la justificación del texto de la leyenda; y la opción `noautotitle` que indica a *gnuplot* que debe incluir en la leyenda únicamente aquellas curvas a las que se ha asignado una etiqueta mediante la opción `title` del comando `plot` (ver última línea del ejemplo, en donde a la primera curva se le da el nombre *Parte real* y a la segunda el nombre *Parte imaginaria* mediante esta opción `title`). Nótese que las figuras de los ejemplos anteriores han sido generados sin esta opción, razón por la cual aparece en la leyenda toda la información dada al comando `plot`.

Las siguientes dos líneas (`set xrange` y `set yrange`) establecen los rangos en *x* e *y* de la gráfica, siguiendo para ello el formato mostrado en el ejemplo. Nótese que la gráfica es idéntica a la del ejemplo 5, habiendo variado los rangos del gráfico y su leyenda, además de las etiquetas y el título.

Por último, las dos líneas que preceden al comando `plot` establecen la manera en que deben presentarse las etiquetas de ordenadas y abscisas. Los comandos `set xtics` y `set ytics` permiten establecer, entre otras cosas, tipos de líneas y fuentes, rotaciones y traslaciones de las etiquetas o marcas (ver `set help xtics` para una descripción completa). Veamos con más detalle una de las opciones más útiles, `autofreq`, mostrada en este ejemplo y que permite establecer un incremento entre etiquetas consecutivas (véase antepenúltima línea, que establece un incremento igual a 0.25), e incluso forzar qué etiqueta aparece en cada punto (véase penúltima línea, donde se establece que aparezcan las etiquetas '0.1 (a)' y '0.4 (b)' en las abscisas 0.1 y 0.4 respectivamente).

## 4.9. Abreviaturas

*Gnuplot* permite abreviar comandos y opciones siempre y cuando dicha abreviatura sea inequívoca. Es decir, que no existen abreviaturas predefinidas, sino que cualquier abreviatura inequívoca será válida, tal y como se ilustra en los ejemplos de la figura 4.8. Algunas de estas abreviaturas se utilizarán en ejemplos posteriores.

## 4.10. Utilización de estilos de líneas y puntos

Otro aspecto importante en relación al formato de una gráfica es el control de formatos de líneas y puntos. Hasta este momento, se ha visto tan solo la opción `with lines`, que permite representar unos datos con trazo continuo en lugar de con puntos, pero ¿cómo puedo seleccionar los colores y formas de las líneas y puntos utilizados para un trazo concreto?

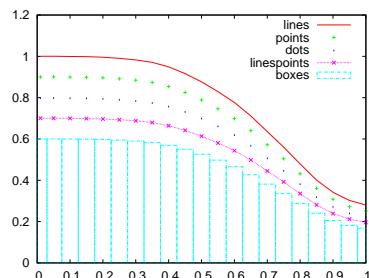
- Ejemplo 1
  - set terminal
  - set termina
  - set termin
  - set term
  - set t
  - se t
- Ejemplo 2
  - set xrange [0:10]
  - set xr [0:10]
- Ejemplo 3
  - plot 'ejemplo.txt' using 1:2 with lines title 'texto determinado'
  - pl 'ejemplo.txt' using 1:2 with lin ti 'texto determinado'
  - p 'ejemplo.txt' u 1:2 w l t 'texto determinado'

**Figura 4.8: Ejemplos de abreviaturas de comandos y opciones.** En cada caso, todas las líneas producen exactamente el mismo resultado

Existen multitud de estilos de trazo disponibles en *gnuplot*. Algunos de ellos están especialmente concebidos para tipologías específicas de gráficos tales como histogramas o funciones de error. La totalidad de estos estilos puede ser consultada a través del comando `help plotting styles`, pero los más usuales (líneas, puntos, puntos pequeños, líneas con puntos y barras) se ilustran en el ejemplo 7.

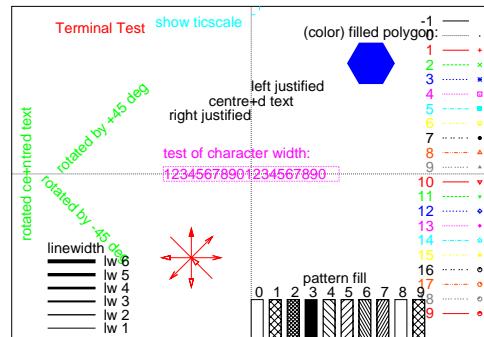
### Ejemplo de gnuplot 7

```
set term postscript enhanced color "Helvetica" 22
set output 'styles.ps'
set yrang [0:1.2]
plot 'ejemplo.txt' u 1:2 w l t 'lines',
     'ejemplo.txt' u 1:(\$2*0.9) w p t 'points',
     'ejemplo.txt' u 1:(\$2*0.8) w d t 'dots',
     'ejemplo.txt' u 1:(\$2*0.7) w lp
     t 'linespoints',
     'ejemplo.txt' u 1:(\$2*0.6) w boxes t 'boxes'
set output
```



**Ejemplo de gnuplot 8**

```
gnuplot> set term postscript
` enhanced color "Helvetica" 22
gnuplot> set output 'test.ps'
gnuplot> test
```



Pero además de elegir si se desea trazar una curva con línea o con puntos, se debe ser capaz de seleccionar características del trazo tales como grosor, color, tipo de línea o de punto, etc.

El primer paso es conocer las posibilidades que ofrece el tipo de terminal con el que se va a trabajar. Para ello, el comando `test` generará un gráfico que ilustra los estilos por defecto, de entre los que es posible seleccionar el más conveniente. El ejemplo 8 ilustra lo que puede obtenerse para una terminal de tipo postscript, si bien hay que tener en cuenta que las posibilidades pueden variar entre diferentes ordenadores.

Una vez conocidas las posibilidades de una terminal, el comando `set style` permite definir estilos concretos. Este comando permite editar las características de una gran variedad de elementos gráficos, tales como líneas, cajas, puntos, vectores, histogramas o etiquetas (ver `help set style`). Vamos a centrarnos aquí en una de sus opciones más necesarias, potentes y utilizadas: `set style line`.

Comparando las ilustraciones de los ejemplos 7 y 8 puede ilustrarse uno de los comportamientos por defecto de *gnuplot*. A la derecha del gráfico de `test` (ejemplo 8) encontramos tres columnas. La primera contiene un número entero, la segunda un trazo y la tercera un punto, siendo cada fila de un color diferente. Cada fila define un estilo de línea por defecto. El estilo 1, en esta terminal y con estas opciones, corresponde a una línea continua o a un signo +, siempre de color rojo. El estilo 8, también de color rojo, tiene un trazo discontinuo (de tipo línea - dos puntos) y el símbolo  $\Delta$ . Fijándonos ahora en la figura del ejemplo 7 puede comprobarse cómo la primera línea (de título 'lines') está trazada acorde al estilo 1, la segunda según el estilo 2, y así sucesivamente.

Los estilos por defecto comentados en el párrafo anterior pueden ser modificados por el usuario, para adecuarlo a sus necesidades, introduciendo el comando

`set style line` seguido del número del estilo que se desea definir y de una o varias opciones elegidas de entre las siguientes:

- `linetype (lt) <tipo de línea>`
- `linecolor (lc) <color>`
- `linewidth (lw) <grosor de linea>`
- `pointtype (pt) <tipo de punto>`
- `pointsize (ps) <tamaño de línea>`

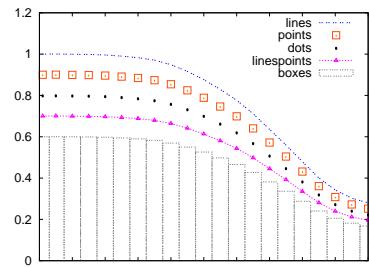
donde entre paréntesis se da la abreviatura más común. Por ejemplo, el comando `set style line 1 linetype 3 linecolor 4 linewidth 1.50 pointtype 2 pointsize 3` define un nuevo estilo 1 de línea con el tipo de trazo del estilo por defecto 3, el color del 4, un espesor de 1.5 puntos y un tamaño de punto de 3 puntos. Este estilo deberá ser invocado con la opción `linestyle` del comando `plot`, cuya abreviatura puede ser `ls`. El ejemplo 9 ilustra el uso de esta posibilidad. Compare el resultado con el ejemplo 7. Nótese también que otra manera interesante de especificar los colores es mediante la opción `linecolor rgb "nombre del color"` (por ejemplo, `linecolor rgb "red"`), donde la lista de nombres de los colores disponibles en cada momento puede obtenerse mediante el comando `show palette colormaps`, con más de 80 colores predefinidos.

### Ejemplo de gnuplot 9

---

```
set term postscript enhanced color
C "Helvetica" 22
set output 'Dstyles.ps'
set yrang [0:1.2]
set style line 10 lt 7 lc 3 lw 2
set style line 2 lc 8 pt 4 ps 2
set style line 3 lc 7 lw 8
set style line 6 lt 3 lc 4 pt 8
set style line 8 lt 9 lc 9
plot 'ejemplo.txt' u 1: w l ls 10 t 'lines',
C 'ejemplo.txt' u 1:($2*0.9) w p ls 2 t 'points',
C 'ejemplo.txt' u 1:($2*0.8) w d ls 3 t 'dots',
C 'ejemplo.txt' u 1:($2*0.7) w lp ls 6 t 'linespoints',
C 'ejemplo.txt' u 1:($2*0.6) w boxes ls 8 t 'boxes'
```

---



## 4.11. Introducción de símbolos en *gnuplot*

La opción `enhanced` del comando `set terminal` permite ampliar las posibilidades del texto introducido en un gráfico. Estas posibilidades dependen en gran medida de cada terminal concreta. Vamos a ver aquí las posibilidades ofrecidas por los gráficos generados en *postscript*, uno de los más potentes en este sentido.

En primer lugar, se dispone de una serie de *modificadores* del texto que nos permiten realizar diversas acciones, resumidas en la tabla 4.1.

Acción	Ejemplo de uso	Resultado
Colocar superíndice	<code>10^{-8}</code>	$10^{-8}$
Colocar subíndice	<code>K_{i,j}</code>	$K_{i,j}$
Alinear sub y superíndices	<code>K@^s_{i,j}</code>	$K_{i,j}^s$
Cambio de fuente	<code>{/Times m}</code>	$m$
Cambio de tamaño	<code>{/=7 m}</code>	$m$
Espacio equivalente	<code>un &amp;{espacio} grande</code>	un grande
Sobreponer dos caracteres	<code>~x{.6-}</code>	$\bar{x}$

Tabla 4.1: Posibilidades adicionales de formateo de texto provistas por la opción `enhanced`

En segundo lugar, existe la posibilidad de introducir una gran variedad de símbolos, recogidos en la figura 4.9 (disponible en el archivo 'ps\_guide.ps' de Richard Crawford). Esta figura muestra una serie de caracteres ordenados en 205 filas y 4 columnas, cada una de las cuales es invocada de manera distinta, como se explica a continuación. La primera columna muestra los caracteres que pueden ser obtenidos directamente en modo texto (en este caso concreto, utilizando tipografía *Times-Roman*). Muchos de estos caracteres pueden ser obtenidos directamente de teclas o combinaciones de teclas del teclado, pero otros no, como por ejemplo el símbolo \$, nº 243. Este símbolo (y cualquier otro) puede obtenerse introduciendo su número entre llaves y precedido de una barra invertida, por ejemplo, '{\243}'.

Los símbolos de la segunda y tercera columnas se obtienen de manera análoga a los de la primera columna, siendo únicamente necesarios establecer la fuente (ver tabla 4.1) correspondiente según lo indicado en la parte superior de la figura. De este modo, el símbolo  $\gamma$  se obtiene mediante '{/Symbol g}' o mediante '{/Symbol \147}', mientras que el símbolo  $\bowtie$  se obtiene mediante '{/ZapfDingbats \053}'.

Por último, para obtener símbolos pertenecientes a la última columna, es necesario seleccionar el conjunto de caracteres denominado ISO Latin1, co-

## PostScript Character Codes

T = text (here Times-Roman) S = Symbol Z = ZapfDingbats E = ISO Latin-1 encoding  
 (the "E" character set is accessed via an option on "set encoding")

T	S	Z	E	T	S	Z	E	T	S	Z	E	T	S	Z	E
040		111	I	I	☆	I		162	r	ρ	□	r	256	fi	→ ③ ®
041	!	! ↗ !	112	J	ø	⊗	J	163	s	σ	▲	s	257	fl	↓ ④ ▴
042	"	∀ ↗ "	113	K	K	★	K	164	t	τ	▼	t	260	°	° ⑤ °
043	#	# ↗ #	114	L	L	★	L	165	u	υ	◆	u	261	-	± ⑥ ±
044	\$	Ξ ↗ \$	115	M	M	★	M	166	v	ω	❖	v	262	†	" ⑦ ²
045	%	% ↗ %	116	N	N	★	N	167	w	ω	►	w	263	‡	≥ ⑧ ³
046	&	& ↗ &	117	O	O	★	O	170	x	ξ		x	264	·	× ⑨ ▾
047	,	϶ ↗ ,	120	P	P	★	P	171	y	ψ	▮	y	265	μ	∞ ⑩ μ
050	(	( ↗ (	121	Q	Θ	★	Q	172	z	ζ	▮	z	266	¶	∂ ① ¶
051	)	) ↗ )	122	R	P	★	R	173	{	{	{	{	267	•	• ② •
052	*	* ↗ *	123	S	Σ	★	S	174					270	,	÷ ③ ,
053	+	+ ↗ +	124	T	T	★	T	175	}	}	"	}	271	„	≠ ④ ¹
054	,	,	125	U	Y	϶	U	176	~	~	"	~	272	"	≡ ⑤ °
055	-	- ↗ -	126	V	ς	★	V	220		1			273	»	≈ ⑥ »
056	.	. ↗ .	127	W	Ω	★	W	221		ˋ			274	... ...	7 ¼
057	/	/ = /	130	X	Ξ	★	X	222		ˊ			275	%o	⑧ ½
060	0	0 ↗ 0	131	Y	Ψ	★	Y	223		^			276	¾ — ⑨ ¾	
061	1	1 ↗ 1	132	Z	Z	⌘	Z	224		~			277	ጀ	↳ ⑩ ገ
062	2	2 ↗ 2	133	[	[	*	[	225		-			300	À	¤ ① À
063	3	3 ↗ 3	134	\	\..	*	\	226		ˇ			301	`	⌚ ② Á
064	4	4 ↗ 4	135	]	]	*	]	227		·			302	‘	⠃ ③ Á
065	5	5 ↗ 5	136	^	⊥	*	^	230		..			303	^	⠃ ④ Ā
066	6	6 ↗ 6	137	—	—	*	—	232		°			304	~	⠃ ⑤ Ä
067	7	7 ↗ 7	140	‘	—	*	‘	233		›			305	-	⊕ ⑥ Å
070	8	8 ↗ 8	141	a	α	*	a	235		ˇ			306	˘	∅ ⑦ Æ
071	9	9 ↗ 9	142	b	β	*	b	236		..			307	·	∩ ⑧ Ç
072	:	: ↗ :	143	c	χ	*	c	237		ˇ			310	..	∪ ⑨ È
073	;	; ↗ ;	144	d	δ	*	d	240		€			311	É	▷ ⑩ É
074	<	< ↗ <	145	e	ε	*	e	241	i	γ	▮	i	312	°	▷ ① É
075	=	= ↗ =	146	f	φ	*	f	242	¢	'	⋮	¢	313	,	▷ ② È
076	>	> ↗ >	147	g	γ	*	g	243	£	≤	⋮	£	314	ì	▷ ③ Í
077	?	? ↗ ?	150	h	η	*	h	244	/	♥	¤		315	~	≤ ④ Í
100	@	≡ ↗ @	151	i	ι	*	i	245	¥	∞	♦	¥	316	„	≡ ⑤ Í
101	A	A ↗ A	152	j	φ	*	j	246	f	f	⊗	!	317	ˇ	⠃ ⑥ Í
102	B	B ↗ B	153	k	κ	*	k	247	§	♣	*	§	320	—	∠ ⑦ Đ
103	C	X ↗ C	154	l	λ	●	l	250	¤	♦	♣	“	321	Ñ	▽ ⑧ Ñ
104	D	Δ ↗ D	155	m	μ	○	m	251	‘	♥	♦	◎	322	Ò	® ⑨ Ò
105	E	E ↗ E	156	n	v	■	n	252	“	♠	♥	^	323	Ó	© ⑩ Ó
106	F	Φ ↗ F	157	o	o	□	o	253	«	↔	♣	«	324	Ô	™ ▶ Ô
107	G	Γ ↗ G	160	p	π	□	p	254	⟨	←	①	¬	325	Õ	Π → Õ
110	H	H ↗ H	161	q	θ	□	q	255	⟩	↑	②	-	326	Ö	√ ↔ Ö
													327	ÿ	ÿ

Figura 4.9: Tabla de símbolos de figuras postscript (Reproducido de 'ps\_guide.ps', elaborado por Richard Crawford)

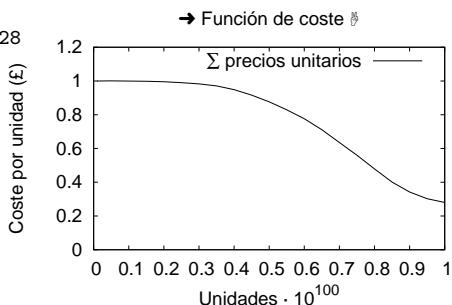
rrespondiente a la escritura de Europa Occidental. Esto se realiza mediante el comando `set encoding iso_8859_1`. El comando `set encoding default` retorna a las fuentes por defecto.

Por último, es importante destacar que la mayoría de las acciones definidas en la tabla 4.1 son fácilmente entendibles tal y como están descritas. Sin embargo, del último modificador (el símbolo ~), utilizado para sobreponer dos caracteres, sí pueden comentarse algunos aspectos extra. En primer lugar, explicar que el número decimal situado entre llaves antes del símbolo a superponer indica el desplazamiento vertical del símbolo a superponer. Así, en la tabla 4.1, a la letra x se le superpone el símbolo –, desplazando previamente éste 0.6 unidades (un 60 % de la altura del texto) en la dirección vertical. Una gran utilidad de este modificador es la posibilidad de introducir caracteres acentuados superponiendo las tildes dadas por los caracteres 301 a 310 de la figura 4.9 a la vocal correspondiente. Así, por ejemplo, el carácter acentuado ó se obtiene introduciendo `~o{\302}`, no siendo necesario especificar ningún desplazamiento vertical de la tilde por estar ésta ya situada a la altura correcta.

El ejemplo 10 muestra la aplicación de todos estos conceptos en un mismo archivo de procedimientos de *gnuplot*.

### Ejemplo de gnuplot 10

```
set term postscript enhanced "Helvetica" 28
set output 'CharactersCodes.ps'
set ylabel 'Coste por unidad (\{\243\})'
set title '{/ZapfDingbats \334}'
 $\hookleftarrow$  Funci~o{\302}n de
 $\hookleftarrow$  coste{/ZapfDingbats \054'}
set xlabel 'Unidades {\264} 10^{100}'
set yrange [0:1.2]
plot 'ejemplo.txt' u 1:2 w l t
 $\hookleftarrow$  {'/Symbol \345} precios unitarios'
```



## 4.12. Gráficos múltiples

En muchas ocasiones resulta interesante, o incluso necesario, generar varios gráficos combinados en una sola imagen, como en los ejemplos mostrados en las figuras 4.2 y 4.3. Esto se consigue utilizando el entorno `multiplot`, denominado entorno porque es necesario establecer cuándo comienza y cuándo termina mediante el par de comandos `set multiplot` y `unset multiplot`. Este entorno puede ser utilizado de dos maneras distintas: permitiendo que *gnuplot* posicione y escale cada uno de los subgráficos dentro del gráfico combinado (opción `layout`), o dejando estas responsabilidades al usuario.

#### 4.12.1. Utilizando la opción layout

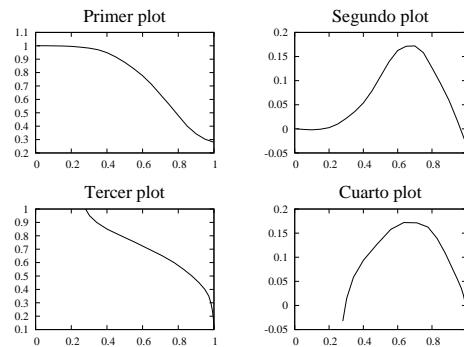
Ilustremos en primer lugar la utilización de la opción `layout` en el ejemplo 11. El fichero de procedimiento ha sido dividido en dos bloques. El primero se limita a establecer la terminal, el fichero de salida, y opciones de leyendas y ejes horizontales. El segundo bloque comprende el entorno `multiplot`, con la opción `layout` seguido de dos número  $m$  y  $n$  separados por comas: el primero estableciendo el número de subgráficos en cada columna, y el segundo estableciendo el número de subgráficos en cada fila, como si cada subgráfico fuese un elemento de una matriz de dimensiones  $m \times n$ . Cada uno de los subgráficos puede estar afectado por variables (rangos, etiquetas, títulos, estilos, etc.) diferentes, pudiendo éstos ser establecidos antes de cada comando tipo `plot`, tal y como se hace en el ejemplo con los títulos individuales.

#### Ejemplo de gnuplot 11

---

```
set term postscript enhanced "Times-Roman" 16
set output 'EjemploMultiplot1.ps'
set key noautotitle
set xr [0:1]

set multiplot layout 2,2
set title '{/=26 Primer plot}'
plot 'ejemplo.txt' u 1:2 w l
set title '{/=26 Segundo plot}'
plot 'ejemplo.txt' u 1:3 w l
set title '{/=26 Tercer plot}'
plot 'ejemplo.txt' u 2:1 w l
set title '{/=26 Cuarto plot}'
plot 'ejemplo.txt' u 2:3 w l
unset multiplot
```




---

Un ejemplo más completo es el mostrado a continuación, donde se transcribe el contenido del fichero de procedimiento utilizado para generar la figura 4.2.

```
set term postscript enhanced color "Times-Roman" 16
set output 'kiu.ps'
set key noautotitle
set key top left
set xr [0:0.25]
set xlabel '{/Italic a_o}'
set encoding iso_8859_1
set format y "%4.1t"
set key center center
```

```

# Definimos los tipos de linea
set style line 1 linetype 1 linecolor rgb "black" linewidth 1.500
set style line 2 linetype 8 linecolor rgb "blue" linewidth 1.500
set style line 3 linetype 2 linecolor rgb "red" linewidth 1.500
set style line 4 linetype 2 linecolor rgb "red" linewidth 1.500
set style line 5 linetype 2 linecolor rgb "blue" linewidth 1.500
set style line 6 linetype 2 linecolor rgb "green" linewidth 1.500
set style line 7 linetype 1 linecolor rgb "gray40" linewidth 1.000
set style line 8 linetype 2 linecolor rgb "gray40" linewidth 1.000

set multiplot layout 2,2

set ylabel "k {\\"327} 10^9"
set yr [0:6]
plot 'k.txt' u 1:($2/1e9) ls 1 w 1 smooth csplines
t '{/Italic K}_{xx}{00/0} (N{/Symbol \244}m)', 'k.txt'
u 1:(-$6/(10*1e9)) ls 2 w 1 smooth csplines t
'{/Italic K}_{x{/Symbol q}}/10&{000} (N),
'k.txt' u 1:($8/(100*1e9)) ls 3 w 1 smooth csplines t
'{/Italic K}_{{}/Symbol q}{/Symbol q}}/100 (N{/Symbol \327}m)'

set ylabel "c {\\"327} 10^{10}"
set yr [0:3]
plot 'k.txt' u 1:($3/($1*1e10)) every ::20 ls 1 w 1 smooth
csplines, 'k.txt' u 1:(-$7/(10*($1*1e10))) every ::20 ls 2 w 1
smooth csplines, 'k.txt' u 1:($9/(100*($1*1e10))) every ::20 ls 3
w 1 smooth csplines

set encoding default
set ylabel "u_b = u/u_{ff}"
set yr [-0.2:1.2]
plot 'iu.txt' u 1:2 ls 1 w 1 smooth csplines t 'Real part',
'iu.txt' u 1:3 ls 2 w 1 smooth csplines t 'Imag part'

set ylabel "{/Symbol q}_b = {/Symbol q}{/Symbol \327}d/u_{ff}"
set yr [-0.002:0.003]
plot 'iu.txt' u 1:4 ls 1 w 1 smooth csplines, 'iu.txt' u 1:5 ls 2
w 1 smooth csplines

unset multiplot

```

Podemos aprovechar este ejemplo para ilustrar tres aspectos interesantes de *gnuplot* que no han sido nombrados hasta el momento. El primero es la espe-

cificación del formato de las etiquetas de los ejes (ver línea 8). El comando `set format`, seguido del eje (x, y ,z) permite especificar la configuración de las cifras presentadas. Los ejemplos de la tabla 4.2 ayudarán a entender el concepto y las diferentes opciones de manera rápida y sencilla. Para evitar que aparezcan, por ejemplo, las etiquetas de ordenadas, puede establecerse un formato vacío, como en `set format y ""`. Más opciones pueden consultarse con el comando `help format`.

Símbolo	Descripción	Ejemplo de uso	Resultados
f	número real	%. %.1	12.1
		%.4f	12.100
		%.0f	12
e	notación exponencial	esto %.0f aparece	esto 12 aparece
		%e %.2e	1.200000e+01 1.20e+01
t,T	mantisa y exponente	%t %T	1.200000 1
		%.2t·10^{ %T}	1.20 · 10 <sup>1</sup>

Tabla 4.2: Ejemplos de formatos en *gnuplot*

En segundo lugar, compárense las líneas 6 y 23. En ambas se configura el texto de una etiqueta. En la línea 6 se hace entre comillas simples ' (tal y como se ha visto hasta ahora), mientras que en la línea 23 se hace entre comillas dobles ". Nótese que ambas opciones son siempre posibles, pero no totalmente equivalentes. La diferencia estriba en que dentro de las comillas dobles, el carácter \ no es reconocido directamente como carácter especial, por lo que hay que ponerlo por duplicado, como en la línea 23. Esto mismo se aplica a otras utilidades del carácter \, como por ejemplo el retorno de carro para una nueva línea, especificado por el texto '\n'.

Por último, en la mayor parte de los comandos `plot` de este ejemplo se usa la opción `with lines smooth csplines`, que une los puntos consecutivos mediante curvas tipo *spline* (Nótese que si no se especifica ningún tipo de opción `smooth`, los puntos se unen mediante líneas rectas, que no siempre generan un resultado visual satisfactorio). Otra opción es `with lines smooth bezier`, que genera una curva de Bezier de grado igual al número de puntos.

#### 4.12.2. Gráficos múltiples sin la opción layout

La opción `layout` nos *asiste* a la hora de generar gráficos múltiples mediante la recolocación y redimensionalización automáticas de los subgráficos. Sin embargo, también el usuario puede hacerse cargo de estas tareas, lo que le confiere mayor control sobre el resultado. Esto se consigue estableciendo el tamaño y la posición de cada uno de los subgráficos mediante los comandos `set size` y `set origin` respectivamente, lo cual caracteriza el área total a ocupar por el gráfico. Sin embargo, este área está ocupada no sólo por el gráfico en sí, sino también por una serie de márgenes que separan unos gráficos de otros y que permiten la colocación de etiquetas y otros elementos. Los valores de dichos márgenes (initialmente calculados de forma automática por *gnuplot*) pueden ser establecidos mediante los comandos mostrados en la tabla 4.3.

El valor numérico del margen puede ser dado directamente después del comando, como por ejemplo `set bmargin 4`, estando las unidades referidas a anchos y altos medios de la fuente en uso. Sin embargo, en muchas ocasiones es útil utilizar la opción `at screen`, que indica la distancia al borde correspondiente del subgráfico en relación con el área total del gráfico completo. Es decir, `set bmargin at screen 0` significa que el borde inferior del subgráfico coincidirá con el borde inferior del área global, mientras que `set lmargin at screen 0.5` indica que el borde izquierdo de los siguientes subgráficos coincidirá con el eje vertical situado justo a la mitad del área imprimible.

El ejemplo 12 ilustra un caso en el que los subgráficos quedan desalineados e irregulares por diversas razones. Posteriormente, el ejemplo 13 muestra cómo puede solucionarse el problema utilizando el comando `set margin` con la opción `at screen`.

Comando	Opciones	Descripción
<code>set bmargin</code>	<code>at screen</code>	Configuración de margen inferior
<code>set tmargin</code>	<code>at screen</code>	Configuración de margen superior
<code>set lmargin</code>	<code>at screen</code>	Configuración de margen izquierdo
<code>set rmargin</code>	<code>at screen</code>	Configuración de margen derecho

Tabla 4.3: Comandos para la configuración de márgenes

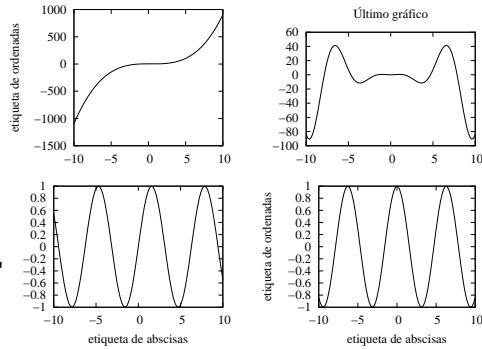
### Ejemplo de gnuplot 12

---

```

set term postscript enhanced "Times-Roman" 18
set output 'MultiplotSinSetMargins.ps'
set key noautotitle
set encoding iso_8859_1
set multiplot
set xlabel 'etiqueta de abscisas'
set origin 0,0
set size 0.5,0.5
plot sin(x)
set ylabel 'etiqueta de ordenadas'
set origin 0.5,0
set size 0.5,0.5
plot cos(x)
unset xlabel
set origin 0,0.5
set size 0.5,0.5
plot x**3-x**2+3
unset ylabel
set title '{\332}ltimo gr{\341}fico'
set origin 0.5,0.5
set size 0.5,0.5
plot cos(x)*x**2
unset multiplot

```



### Ejemplo de gnuplot 13

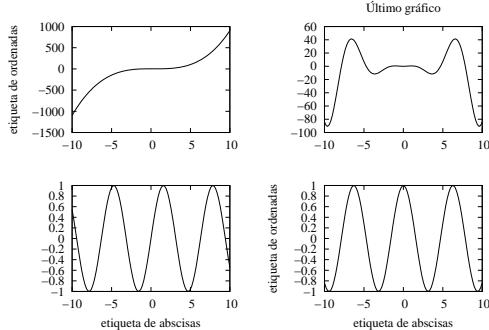
---

```

set term postscript enhanced "Times-Roman" 18
set output 'MultiplotSetMargins.ps'
set key noautotitle
set encoding iso_8859_1

set multiplot
set xlabel 'etiqueta de abscisas'
set bmargin at screen 0.15
set tmargin at screen 0.45
set lmargin at screen 0.15
set origin 0,0
set size 0.5,0.5
plot sin(x)
set ylabel 'etiqueta de ordenadas'
set lmargin at screen 0.65
set origin 0.5,0
set size 0.5,0.5
plot cos(x)
set bmargin at screen 0.6
set tmargin at screen 0.9
set lmargin at screen 0.15
unset xlabel
set origin 0,0.5
set size 0.5,0.5
plot x**3-x**2+3
unset ylabel
set lmargin at screen 0.65

```



```
set title '{\332}ltimo gr{\341}fico'
set origin 0.5,0.5
set size 0.5,0.5
plot cos(x)*x**2
unset multiplot
```

---

Para terminar, un ejemplo más completo es el mostrado a continuación, donde se transcribe el contenido del fichero de procedimiento utilizado para generar la figura 4.3.

```
set term postscript enhanced color "Times-Roman" 18
set output 'AxialForce.ps'
set encoding iso_8859_1
set ylabel 'depth (m)'
set xrange [0:2e8]
set xlabel "Axial force / u_f      (N/m)"
set key right top

set style line 1 lt 1 lc rgb "black" lw 1.30 pt 1
set style line 2 lt 1 lc rgb "blue"   lw 1.30 pt 2
set style line 3 lt 1 lc rgb "red"    lw 1.30 pt 4
set style line 4 lt 1 lc rgb "green"  lw 1.30 pt 6
set style line 5 lt 8 lc rgb "black" lw 1.30 pt 1
set style line 6 lt 8 lc rgb "blue"  lw 1.30 pt 2
set style line 7 lt 8 lc rgb "red"   lw 1.30 pt 4
set style line 8 lt 8 lc rgb "green" lw 1.30 pt 6

set multiplot
set bmargin 3.2
set origin 0,0
set size 0.68,1
set xtics autofreq ("0" 0, "0.5{/Symbol \327}10^8" 0.5e8,
"1{/Symbol \327} 10^8" 1.0e8, "1.5{/Symbol \327}10^8" 1.5e8,
"2{/Symbol \327}10^8" 2e8)
plot 'fichero1.txt' u 3:9 [...]

set origin 0.6,0
set size 0.4,1
set key off
set xlabel '{/Italic a}_o'
set xr [0:0.3]
unset ylabel
set format y ""
```

```
set xtics autofreq ("0.1" 0.1, "0.2" 0.2, "0.3" 0.3)
plot 'fichero1.txt' u 4:9 [...]
unset multiplot
```

## 4.13. Funciones implementadas en *gnuplot*

*Gnuplot* dispone de funciones matemáticas ya implementadas. La totalidad de dichas funciones, y detalles sobre las mismas, pueden ser consultadas, por ejemplo, en [7] y en el propio *gnuplot* utilizando el comando `help functions`. La tabla 4.4 presenta un resumen de las funciones más utilizadas.

<code>sin(x)</code> , <code>cos(x)</code> , <code>tan(x)</code> , <code>asin(x)</code> , <code>acos(x)</code> , <code>atan(x)</code> , <code>sinh(x)</code> , <code>cosh(x)</code> , <code>tanh(x)</code> , <code>asinh(x)</code> , <code>acosh(x)</code> , <code>atanh(x)</code>	funciones trigonométricas y parabólicas, incluidas sus inversas
<code>exp(x)</code>	Función $e^x$
<code>log(x)</code> , <code>log10(x)</code>	Logaritmos natural (base $e$ ) y decimal (base 10)
<code>sqrt(x)</code>	Raíz cuadrada
<code>besj0(x)</code> , <code>besy0(x)</code> , <code>besj1(x)</code> , <code>besy0(x)</code>	Funciones de Bessel
<code>abs(x)</code>	Valor absoluto
<code>sgn(x)</code>	si $x > 0 \Rightarrow \text{sgn}(x) = 1$ si $x < 0 \Rightarrow \text{sgn}(x) = -1$ si $x = 0 \Rightarrow \text{sgn}(x) = 0$

Tabla 4.4: Algunas de las funciones implementadas en *gnuplot*

## 4.14. Operadores y condicionales en *gnuplot*

### 4.14.1. Operadores de dos argumentos

*Gnuplot* dispone de una serie de operadores con formato muy cercano a los pertenecientes a lenguajes de programación como *C* o *Fortran*. La tabla 4.5 muestra los operadores ordinarios de dos argumentos, cuyo funcionamiento es el usual, mientras la tabla 4.6 muestra los operadores de resultado binario (uno o cero). El uso de estos operadores se ilustra en el ejemplo 14, donde se aprecia que el resultado de la operación es 1 o 0 en función de que la condición se cumpla o no. La tabla 4.7 muestra los operadores binarios, que necesitan argumentos de tipo entero. Estos operadores no pueden operar directamente

Símbolo	Ejemplo de uso	Descripción
**	a**b	potencia
*	a*b	multiplicación
/	a/b	división
+	a+b	suma
-	a-b	resta

Tabla 4.5: Lista de operadores binarios ordinarios en *gnuplot*

Símbolo	Ejemplo de uso	Descripción
==	a==b	igualdad
!=	a!=b	desigualdad
<	a<b	menor que
<=	a<=b	menor o igual que
>	a>b	mayor que
>=	a>=b	mayor o igual que

Tabla 4.6: Lista de operadores binarios de *resultado binario* en *gnuplot*

Símbolo	Ejemplo de uso	Descripción
%	a % b	módulo
^	a ^ b	OR exclusivo
	a   b	OR inclusivo
&&	a && b	AND lógico
	a    b	OR lógico

Tabla 4.7: Lista de operadores binarios para *argumentos exclusivamente de tipo entero* en *gnuplot*

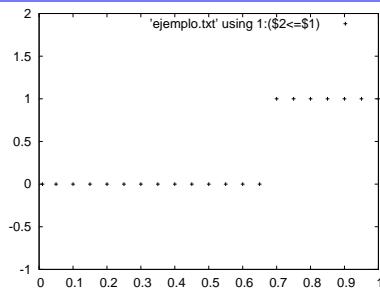
Símbolo	Ejemplo de uso	Descripción
.	A.B	Concatenación
eq	A eq B	Igualdad
ne	A ne B	Desigualdad

Tabla 4.8: Lista de operadores binarios para *argumentos exclusivamente de tipo cadena de caracteres* en *gnuplot*

con datos leídos de un fichero de datos, aunque sí pueden operar con funciones definidas por el usuario. Sin embargo, sí pueden realizarse operaciones como, por ejemplo, `plot 'ejemplo.txt' using 1:((\$2<=$1)&&(\$3<$2))*3` que dará como resultado puntos en 0 o en 3 en función de que se cumpla ambas condiciones simultáneamente o no. Por último, la tabla 4.8 muestra las operaciones que pueden realizarse entre cadenas de caracteres, utilizadas sobre todo para títulos y etiquetas.

#### Ejemplo de gnuplot 14

```
gnuplot> plot 'ejemplo.txt'
      < using 1:(\$2<=$1)
```



#### 4.14.2. Operadores de un único argumento

Además de los operadores vistos en el apartado anterior, *gnuplot* ofrece operadores de un solo argumento, entre los que destacan el cambio de signo, la negación lógica y el cálculo del factorial (con argumento entero) tal y como se muestra en la tabla 4.9,

Símbolo	Ejemplo de uso	Descripción
-	-a	Cambio de signo
!	!a	Negación lógica
!	a!	Factorial

Tabla 4.9: Lista de operadores unitarios en *gnuplot*

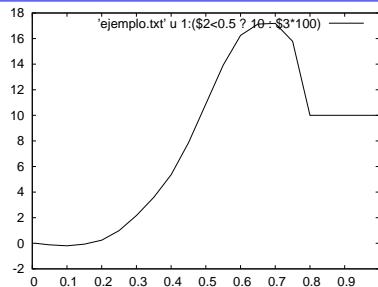
#### 4.14.3. Condicionales

Por último, *gnuplot* facilita el uso de condicionales mediante lo que denomina 'operador ternario', que permite la evaluación de condiciones del tipo `if then else`, a través de la sintaxis `a?b:c`. Aquí, el primer argumento (a, de tipo entero o lógico) es evaluado. En caso de tener valor mayor que cero

(true), el segundo argumento (b) es evaluado y su valor es el retornado por el condicional. Si  $a=0$ , entonces se evalúa y devuelve la parte de la sentencia situada después de los dos puntos (c), de modo que el símbolo : podría entenderse como un *sino*. El ejemplo 15 muestra una aplicación sencilla (fíjese cómo, a partir de  $x = 0.8$ , cuando la condición  $\$2 < 0.5$  comienza a ser cierta,  $f(x)$  toma el valor 10).

### Ejemplo de gnuplot 15

```
gnuplot> plot 'ejemplo.txt' using  
1:(\$2<0.5 ? 10 : \$3*100) with lines
```



## 4.15. Gnuplot y LATEX

### 4.15.1. Haciendo que sea LATEX quien procese el texto y los símbolos de una imagen: el terminal epslatex

Uno de los posibles destinos de una representación gráfica generada con *gnuplot* es su inclusión en un documento LATEX. Para ello, una opción obvia y, en muchas ocasiones, satisfactoria, es la generación de una imagen, ya sea de tipo .ps o .eps si se va a compilar con LATEX, o de tipo .png o .pdf si se va a compilar con PDFLATEX. En este caso, todos los elementos, incluyendo textos y símbolos, quedan totalmente definidos en la imagen y no pueden ser modificados por LATEX.

Sin embargo, en muchas ocasiones resulta interesante que pueda ser el propio LATEX el que procese y genere el texto y los símbolos de una representación gráfica. Esto conlleva, principalmente, dos beneficios:

- Asegurar la homogeneidad entre las fuentes utilizadas en el texto del documento y el texto de la figura, y
- Generar símbolos y ecuaciones utilizando un lenguaje y nomenclatura coherente con el resto del documento, y pudiendo hacer uso de cualquier recurso de LATEX.

Esto último se consigue con la utilización de la terminal `epslatex`. La mayor parte de las terminales (`png`, `jpeg`, `postscript`, etc.) generan un único archivo de salida. La terminal `epslatex`, por el contrario, genera dos archivos:

1. Un archivo `.eps` que contiene, exclusivamente, los elementos gráficos de la figura, y
2. Un archivo `.tex` que contiene las instrucciones para que  $\text{\LaTeX}$  procese e introduzca el texto correcto en la figura.

Para ello, la terminal necesita que el archivo de salida definido con el comando `set output` tenga extensión `.tex`, generando también un archivo `.eps` con el mismo nombre de archivo y modificando únicamente la extensión. Así, por ejemplo, el fichero de procedimiento

**fichero de texto `grafico-en-epslatex.gnuplot`**

```
set term epslatex size 8cm,5.5cm
set output 'MiPrimerEpslatex.tex'
set key noautotitle
set xr [0:5]
set label 'Ejemplo de eps\textrm{\LaTeX}' at 0.5,12
set xlabel '$x$'
set ylabel '$\displaystyle\int\frac{x+1}{\sqrt{x-1}}$'
plot (2*(x-1)/3+4)*sqrt(x-1)
set output
```

generará **dos** archivos: `MiPrimerEpslatex.tex` y `MiPrimerEpslatex.eps`. Posteriormente, para introducir la representación gráfica generada con `gnuplot` en un documento  $\text{\LaTeX}$  se introduciría `\input{MiPrimerEpslatex.tex}`. Así, en un documento  $\text{\LaTeX}$ , el código

```
\begin{figure}[h]
\centering
\input{ilustraciones/CapGnuplot/MiPrimerEpslatex.tex}
\end{figure}
```

genera la figura 4.10 (a falta, por supuesto, de la definición del pie de figura y etiqueta). Nótese también que cuando un elemento deba ser interpretado en modo matemático, deberá escribirse entre dos signos `$`, tal y como se haría en un documento  $\text{\LaTeX}$  (ver sección 2.4.6).

#### 4.15.2. *Gnuplot, beamer y fondos transparentes en una ilustración*

En el capítulo 2 se nombró el paquete `beamer` de  $\text{\LaTeX}$  como una buena opción para la generación de presentaciones. Tanto si se utiliza `beamer` como si se

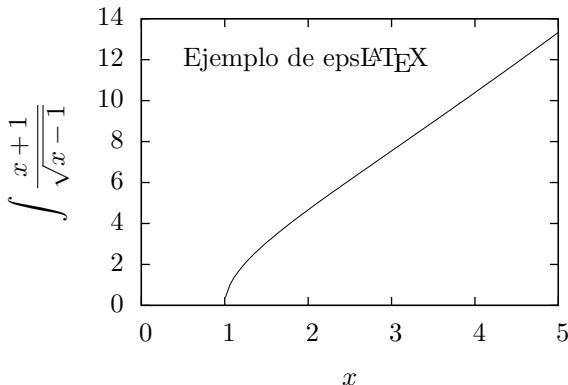


Figura 4.10: Ejemplo de figura generada con la terminal `epslatex` de `gnuplot`

utiliza alguna otra herramienta, es muy común (e incluso recomendable) evitar un fondo blanco para las presentaciones, entre otras cosas, porque cansará la vista de los asistentes con mucha mayor rapidez que en caso de elegir un fondo oscuro. En todo caso, y si el fondo no es estrictamente blanco, será deseable producir una figura transparente, de modo que al introducir la figura en la presentación, se conserve el color del fondo.

La mejor opción para generar figuras transparentes, más aún si su destino final es `beamer`, es utilizar la terminal `png` con las opciones `transparent`, `font`, `enhanced` y `size`. Con la opción `font` se ha elegido un fuente `serif` de tamaño 60 puntos, acorde al tamaño, sensiblemente grande, definido con la opción `size`. Este tamaño, mucho mayor del tamaño por defecto ( $640 \times 480$ ), se elige para aumentar la resolución de la imagen. Por otro lado, la opción `enhanced` permite el uso de las opciones adicionales de formato de texto indicados en la tabla 4.1.

La mayoría de las líneas siguientes utilizan comandos ya conocidos para establecer el fichero de salida, el título, etiquetas, estilo de líneas, etc. El elemento novedoso es la opción `textcolor` para los comandos `set title`, `set key`<sup>5</sup>, `set label` y `set xlabel`, con el que puede definirse el color del texto, en este ejemplo el blanco a través de su nombre RGB. El único comando nuevo es `set`

---

<sup>5</sup> En el caso del comando `set key`, la opción `textcolor` sólo está disponible a partir de la versión 4.4 de `gnuplot`.

border, con el que es posible definir diversas propiedades de los bordes de la figura (ver `help set border` para una información completa al respecto). La opción `31` permite controlar todos los bordes con un solo comando, y con la opción `linestyle (ls)`, combinada con una definición previa de un estilo de línea con el comando `set style`, se configuran todos los bordes con el color blanco.

fichero de texto `grafico-en-png-fondo-transparente.gnuplot`

```
set term png transparent font serif 60 enhanced size 2560,1920
set output 'png-blanco-sobre-transparente.png'
set title 'Ejemplo de png transparente' textcolor rgb 'white'
set style line 1 linecolor rgb 'white' lw 4
set border 31 ls 1
set key textcolor rgb 'white'
set label 1 'Ejercicio de gnuplot' at 0.1,2 textcolor rgb 'white'
set xlabel 'x' textcolor rgb 'white'
set ylabel 'sin(x)' textcolor rgb 'white'
plot sin(x) ls 1
set output
```

La figura generada con este fichero, e introducida en una presentación con fondo gris, generaría el resultado mostrado en la figura 4.11, menos agresiva para la audiencia que una figura en líneas negras sobre un fondo blanco cuando se está utilizando un proyector, sobretodo cuando se apagan las luces del auditorio.

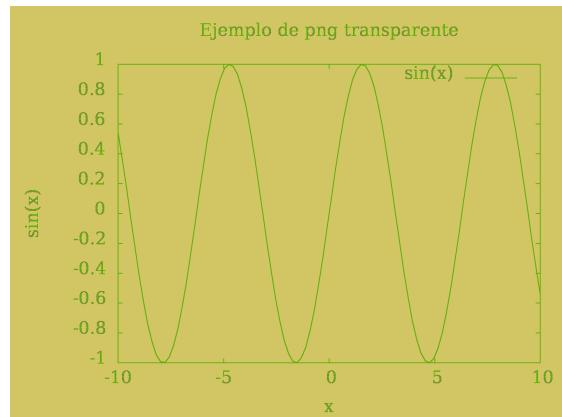


Figura 4.11: Ejemplo de figura transparente generada con la terminal png de gnuplot

## CAPÍTULO 5

### Control de versiones con *subversion*

## 5.1. ¿Qué es y para qué sirve un sistema de control de versiones?

Un documento o programa de cierta magnitud, escrito en solitario o en colaboración con otros compañeros, puede conllevar semanas, meses o incluso años de trabajo durante los que el proyecto atraviesa distintas fases. Guardar innumerables copias, con distintas fechas, de las carpetas o ficheros de trabajo, podría ser una opción para evitar pérdidas de información y gestionar las aportaciones de distintas personas. Sin embargo, a la hora de enfrentarse a proyectos de cierta envergadura (una tesis doctoral, un proyecto de ingeniería o de arquitectura o un programa de cierta complejidad) se hace deseable tener la posibilidad controlar los distintos estados por los que va transcurriendo el proceso. También puede ser útil guardar un *registro* (a modo de diario) de los cambios que se van realizando y, en su caso, gestionar las aportaciones de distintos participantes. De este modo, podría recuperarse o consultarse un determinado contenido tal y como existía en un determinado momento. Y obviamente, todo lo que se ha comentado aquí debería realizarse de manera preferentemente automatizada y ordenada, sin necesidad de generar múltiples copias del mismo documento, y con la posibilidad de acceder al sistema desde distintos puestos de trabajo.

Estas necesidades (y otras que veremos más adelante) quedan plenamente cubiertas a través del uso de *sistemas de control de versiones*, inicialmente ideados para gestionar el trabajo colaborativo de equipos de programadores, pero ampliamente utilizados hoy en día también en la redacción de documentos, no sólo a nivel colaborativo sino también individual.

En estos momentos existen varios sistemas de control de versiones ampliamente utilizados y contrastados, siendo más recomendables unos u otros en función de la aplicación concreta. En este capítulo se introduce el uso de uno de ellos: *subversion*, capaz de satisfacer las necesidades que puedan surgir durante la elaboración y revisión de todo tipo de documentos científicos y técnicos. Entre otras ventajas, y como se verá más adelante, *subversion* dispone de diversas interfaces gráficas que pueden hacer la tarea aún más sencilla y que lo hacen más atractivo que otras alternativas.

## 5.2. *Subversion*: un sistema de control de versiones

*Subversion* es un programa libre y de código abierto que permite gestionar la *historia* de ficheros y directorios, recordando y organizando el contenido en cada momento, los cambios que han sufrido, y quién y porqué los ha realizado. Se comporta, en cierto modo, como una máquina del tiempo, con la característica adicional de que los cambios quedan documentados en mayor o menor medida, de modo que si fuera necesario es posible recuperar el contenido de

un documento cualquiera en un instante de tiempo dado, o revisar cómo se ha ido desarrollando un proyecto.

Además, *subversion* trabaja tanto con ficheros de texto como con ficheros binarios (imágenes, vídeos, ejecutables, . . . ), además de con directorios y propiedades, lo que permite aplicarlo a casi cualquier actividad, desde la redacción de un texto en solitario hasta la escritura de un libro o de un programa en colaboración con múltiples compañeros.

### 5.3. ¿Interferirá *subversion* en mi trabajo diario?

Aún no se ha visto en qué se concreta todo esto, por lo que probablemente usted se estará haciendo preguntas del tipo ¿cómo va a interferir *subversion* en mi trabajo diario? ¿Podré seguir usando las mismas herramientas y programas que uso hasta ahora para hacer mis tareas? ¿Hacer uso de un sistema de control de versiones me va a dificultar aún más el día a día? ¿Afectará a todo mi trabajo?

*Subversion* no va a interferir en absoluto en su trabajo, ni va a condicionar en modo alguno las herramientas o los programas que utilice para sus tareas (podrá seguir usando sus programas preferidos). *Subversion* se limitará, exclusivamente, a gestionar el contenido de los trabajos que usted indique, en el momento y forma que el usuario establezca. Así, usted sólo necesitará dedicar unos instantes a la tarea de gestionar el proyecto.

### 5.4. Obtención e instalación de *subversion*

La página oficial de *subversion* es <http://subversion.apache.org/>. En el menú de la izquierda se dispone de los enlaces Binary Packages y Source Code para obtener ejecutables o códigos fuente del programa. Haciendo clic en Binary Packages se tiene acceso a diversas opciones para los diferentes sistemas operativos.

En el caso del sistema operativo  Windows, la opción más recomendable es la ofrecida por *CollabNet*, que permite descargar la versión original en un archivo del tipo: CollabNetSubversion-server-\*.exe. Sin embargo, en el caso de  GNU/Linux, *subversion* está por lo general disponible directamente en los repositorios de la mayoría de las distribuciones (para instalarlo en Ubuntu o en Debian, puede utilizarse 'Synaptic' o puede teclearse en una terminal sudo apt-get install subversion). Por último, en el caso de  Mac OS X, *subversion* forma parte las Apple's Developer Tools disponibles, generalmente, entre el software vendido junto al equipo, y en todo caso desde la página web del fabricante. También puede instalarse a través de *fink*.

*Subversion*, al igual que *gnuplot*, es un programa en línea de comandos sencillo, potente y que puede utilizarse de manera totalmente autónoma desde la ter-

minal. Así lo utiliza generalmente el autor de este texto. Sin embargo, existen diversos programas, distintos del propio *subversion*, que ofrecen una interfaz gráfica para comunicarse con él y cuya intención es facilitar, aún más si cabe, su uso. A lo largo del capítulo se ilustrará el uso de dos de ellos: *RapidSVN*, que se puede obtener desde su página web (<http://rapidsvn.tigris.org/>) o directamente de los repositorios en el caso de  $\Delta$  GNU/Linux; y *TortoiseSVN*<sup>6</sup>, descargable desde su página web (<http://tortoisessvn.tigris.org/>) y exclusivo para el sistema operativo  $\blacksquare$  Windows. Ambos son proyectos de software libre, bajo licencia GNU General Public License, pero muy distintos entre sí. Por un lado, *TortoiseSVN* está integrado con el explorador de archivos y los menús contextuales del sistema operativo<sup>7</sup> (ver figura 5.1), razón por la cual sólo está disponible para  $\blacksquare$  MS Windows. Por el contrario, *RapidSVN*, que está disponible para los tres sistemas operativos, funciona de manera independiente del explorador de archivos (ver figura 5.2).

Los conceptos y las órdenes principales para ambos programas son esencialmente idénticas a las utilizadas desde la línea de comandos. Por eso, aprender las ideas principales y el vocabulario básico permitirá posteriormente usar *subversion* en línea de comandos o desde cualquier interfaz gráfica.

## 5.5. Conceptos básicos y flujo de trabajo

La figura 5.3 muestra de manera esquemática el ciclo de trabajo que se sigue generalmente en el uso de programas como *subversion*. El camino en color rojo muestra las acciones básicas y más comunes, siendo el resto acciones siempre opcionales o de uso avanzado y no tan frecuente.

Describiremos a continuación las acciones y los conceptos más importantes para el manejo de *subversion*. Los siguientes apartados explican dichos conceptos junto con los comandos originales del programa. Aunque, en la mayoría de los casos, y por simplicidad, los comandos están inicialmente expuestos para ser utilizados en línea de comandos, tenga en cuenta que los conceptos y las órdenes son exactamente los mismos también si va a utilizar una interfaz gráfica. Una vez entendido el concepto y vista la orden, usted podrá realizar cada acción directamente desde *subversion*, o a través de *TortoiseSVN* o *RapidSVN*.

---

<sup>6</sup> Para utilizar *TortoiseSVN* no es necesario instalar previamente el programa de *CollabNet* para su uso en línea de comandos. Por el contrario, *RapidSVN* requiere la instalación de *subversion* en cualquier sistema operativo.

<sup>7</sup> Para  $\Delta$  GNU/Linux, *RabbitVCS* es un magnífico programa con un enfoque muy similar al de *TortoiseSVN*.

## Control de versiones con subversion



Figura 5.1: Pantallazo de la interfaz *TortoiseSVN*, sólo disponible para Windows

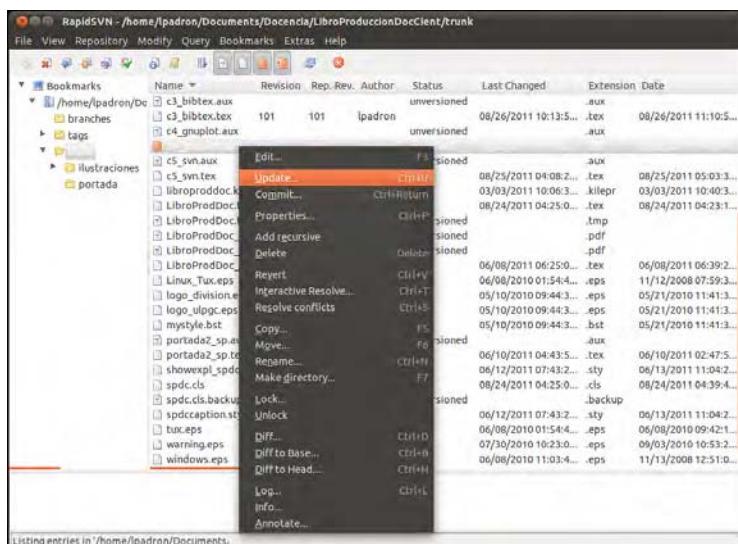


Figura 5.2: Pantallazo de la interfaz *RapidSVN*, disponible en todos los Sistemas Operativos

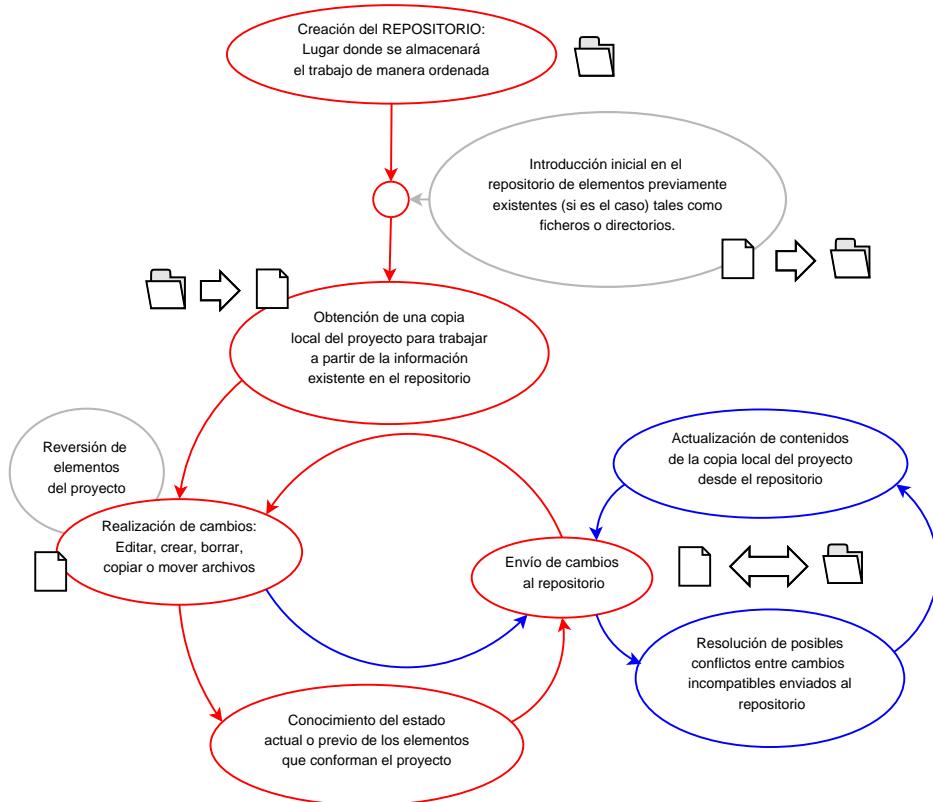


Figura 5.3: Ciclo de trabajo en el uso de un sistema de control de versiones como *subversion*

### 5.5.1. Creación de un repositorio

El primer paso es construir el almacén central donde *subversion* almacenará, de manera organizada, los elementos que se desee poner bajo control de versiones. Este almacén recibe el nombre de *Repositorio*, y se puede crear mediante el comando `svnadmin create8`. Los siguientes son ejemplos de creación de repositorios<sup>9</sup>:

<sup>8</sup> El usuario nunca trabajará directamente en el repositorio, sino en sus directorios habituales. Posteriormente pedirá a *subversion* que se comunique con el repositorio y realice las tareas oportunas. De hecho, si crea un repositorio y mira su contenido, verá que se ha generado una serie de archivos y directorios opacos para el usuario. Pero no se preocupe de eso, nunca deberá modificar directamente el contenido de un repositorio.

<sup>9</sup> Note que la barra a utilizar para denotar los directorios es siempre /, incluso cuando se trabaje en Windows.

```
$ svnadmin create MiRepo  
$ svnadmin create /home/lpadron/MiRepo  
$ svnadmin create "C:/Documents and Settings/Luis Padron/  
Mis documentos/MiRepo"
```

Para empezar, consideraremos que el trabajo va a ser almacenado localmente en el propio ordenador del usuario. Más adelante se comentarán otras posibilidades.

### 5.5.2. *Sobre la estructura recomendada de un repositorio: tags y branches*

Aunque no es obligatorio, y al principio no parezca necesario, en la mayoría de los casos es interesante dar al repositorio una estructura inicial de tres carpetas denominadas *trunk*, *tags* y *branches* que puede facilitar el trabajo en el futuro. En la primera de las carpetas (*trunk*) se suele almacenar el trabajo principal en desarrollo (es decir, el día a día); la segunda (*tags*), contendría *momentos* o estados de especial interés del trabajo (como por ejemplo el primer borrador del documento o la versión 2.0 del programa); y en la tercera (*branches*) se trabajaría (si es el caso) en distintas ramas o variaciones del proyecto. Este último es el caso en el que interesa tener dos o más variantes de un mismo trabajo en el que partes importantes de estas variantes siguen guardando relación, de manera que el trabajo de una variante pueda ser aprovechado por el resto (erratas localizadas en una variante del documento que interesa que sean corregidas en todos los casos, errores encontrado en un programa que afecta a todas sus versiones, etc.).

Puede crearse esta estructura directamente en el repositorio con el comando `svn mkdir`. Por ejemplo, en Mac OS X o en GNU/Linux, desde una terminal, se introduciría:

```
$ svn mkdir file:///home/lpadron/MiRepo/trunk --message "creando  
directorio principal"  
  
$ svn mkdir file:///home/lpadron/MiRepo/tags --message "creando  
directorio para guardar estados determinados del proyecto"  
  
$ svn mkdir file:///home/lpadron/MiRepo/branches --message "creando  
directorio para desarrollo de posibles variantes"
```

donde, `file:///home/lpadron/MiRepo` es la dirección completa en formato URL del repositorio. Si, como hasta ahora, el repositorio reside en el mismo ordenador en el que se va a trabajar, la URL es la dirección completa del repo-

sitorio precedida por `file://`. En el caso de repositorios en red, se verá el formato más adelante.

En el caso de Windows, y desde una terminal, los comandos serían idénticos, teniendo únicamente que cuidar el formato de las URLs:

```
$ svn mkdir "file:///C:/Documents and Settings/Luis Padron/Mis  
documentos/MiRepo/trunk" --message "creando directorio principal"  
  
$ svn mkdir "file:///C:/Documents and Settings/Luis Padron/  
Mis documentos/MiRepo/tabs" --message "creando directorio para  
guardar estados determinados del proyecto"  
  
$ svn mkdir "file:///C:/Documents and Settings/Luis Padron/  
Mis documentos/MiRepo/branches" --message "creando directorio  
para guardar estados determinados del proyecto"
```

Finalmente, fíjese que la estructura inicial, y los nombres de las carpetas, son totalmente flexibles, y usted puede decidir introducir muchas o ninguna carpeta en lugar de 3, y en su caso puede darles el nombre más significativo para su caso. De hecho, usted puede preferir no crear carpetas ninguna de esta forma, con lo que obviaría este punto y continuaría en el siguiente. Esta estructura inicial de tres carpetas es, por así decirlo, una recomendación general derivada de una forma concreta de trabajar.

Por supuesto, e independientemente de esta estructura inicial recomendada, usted podrá crear posteriormente, si así lo desea, todas las carpetas, subcarpetas y archivos que considere necesarios para el desarrollo de su proyecto.

### 5.5.3. *Obtención de una copia local de trabajo*

El almacén ya está construido. A partir de ahora, se almacenará aquí todo el trabajo, y alguien llevará y traerá los documentos según vaya haciendo falta. Sin embargo, no se trabaja en el almacén, sino en el lugar de trabajo preferido. Hace falta, por tanto, definir ahora cuál es el lugar de trabajo, relacionándolo al mismo tiempo con el repositorio recién creado.

Esta operación de definir el lugar de trabajo, que podríamos llamar *generación de una copia local de trabajo*, se realiza mediante el comando `svn checkout`, tal y como se ve en los ejemplos siguientes:

```
$ svn checkout file:///home/lpadron/MiRepo/trunk MiCopiaDeTrabajo  
$ svn checkout file:///home/lpadron/MiRepo MiCopiaDeTrabajo  
$ svn checkout "file:///C:/Documents and Settings/Luis Padron/  
Mis documentos/MiRepo/trunk" MiCopiaDeTrabajo
```

donde tras el comando se facilita, en primer lugar, la dirección del repositorio, y en segundo lugar la dirección y el nombre de la copia de trabajo. En este caso, se creará el directorio *MiCopiaDeTrabajo* (sobra decir que puede tener cualquier nombre) donde el usuario puede ya empezar a trabajar.

En el segundo ejemplo se extrae todo lo contenido en el repositorio (si se hubiese preparado la estructura *trunk*, *tags* y *branches*, se obtendría todo). En el primer y tercer ejemplos se extrae, en su caso, tan solo lo contenido en *trunk*, es decir, la línea principal de desarrollo.

#### 5.5.4. Información de la copia local de trabajo

Para obtener, entre otra información de interés, la URL del repositorio asociado a una copia de trabajo, se utiliza el siguiente comando:

```
$ svn info
```

#### 5.5.5. Trabajo: creación, edición, copia y eliminación de archivos y carpetas

Hasta aquí los preparativos previos que, por otro lado, no deberían llevar más allá de 2 o 3 minutos: se ha construido el almacén, se ha definido, quizás, una estructura, y finalmente se ha establecido un lugar de trabajo. Ahora queda lo más importante: trabajar. Veamos ahora las operaciones básicas a utilizar en el día a día.

Situémonos dentro de la copia de trabajo. Desde el punto de vista del usuario, ésta será una carpeta normal y corriente, como las demás. Se puede entrar en ella, trabajar en ella, copiar archivos a o desde ella, guardar archivos nuevos, crear carpetas, etc., pero cualquier elemento creado de manera *normal* no estará supervisado por *subversion* hasta que, de una manera u otra, se indique que así sea.

Partiendo de esta idea, consideremos ahora un archivo *MiDocumento.tex* situado en la carpeta de trabajo. Puede que haya sido copiado desde otro lugar, o que haya sido guardado o creado directamente ahí. En cualquier caso, si forma parte del proyecto y se desea colocarlo bajo control de versiones, el comando sería el siguiente:

```
$ svn add MiDocumento.tex
```

Si se ha creado una carpeta llamada *ilustraciones* para guardar las imágenes, la carpeta, y todo su contenido, se añade del mismo modo:

```
$ svn add ilustraciones
```

En este caso, si sólo se desea añadir la carpeta y no su contenido (por ejemplo, porque dentro hay muchos archivos auxiliares o no útiles que no se desean poner bajo control de versiones), entonces la opción sería la siguiente:

```
$ svn add --depth empty ilustraciones
```

En cuanto a la creación de elementos, se puede crear una nueva carpeta, colocándola directamente bajo la supervisión de *subversion*, con el siguiente comando:

```
$ svn mkdir ilustraciones
```

Consideremos ahora que se desea crear un nuevo archivo a partir de uno existente, y que además interesa que dicha acción se recuerde, para tener constancia de que el nuevo archivo proviene de otro ya existente y bajo control de versiones. En este caso, el comando sería:

```
$ svn copy MiDocumento.tex CopiaDeMiDocumento.tex
```

En lugar de copiarlo, puede interesar cambiarle el nombre, pero de modo que quede constancia de esto y se mantenga todo su historial anterior. Para ello, tan solo hay que ejecutar:

```
$ svn move MiDocumento.tex NuevoNombre.NuevaExtension
```

Y por supuesto, es muy posible que en algún momento haya archivos que ya no sean necesarios. Para eliminarlos<sup>10</sup> de la copia de trabajo y de la próxima versión residente en el repositorio, se introduciría:

```
$ svn delete ElementoParaBorrar
```

### **5.5.6. Envío de los cambios al repositorio**

Ahora que ya hemos trabajado duro durante algunas horas en las que hemos creado y modificado archivos y carpetas, llega el momento de *enviar* el nuevo material al repositorio. Para ello, situados en la carpeta principal de la copia de trabajo, se introduce:

```
$ svn commit --message "Comentarios sobre los cambios realizados"
```

---

<sup>10</sup> En realidad, nunca se borra nada de *subversion*. Al ser un sistema de control de versiones, podemos decir que el archivo ya no aparecerá en la revisión actual (ver sección 5.5.9) pero su historia seguirá ahí y podrá ser recuperado, en cualquier momento, al estado en que se encontraba dicho archivo en cualquier fecha anterior.

Con la acción `commit` se envían todos los cambios al repositorio. *Subversion obliga* a documentar estos cambios con la introducción de un texto (otra cosa, por supuesto, es que el usuario haga o no buen uso de ellos). Si no se incluye la opción `--message`, *subversion* pedirá los comentarios antes de enviar los cambios. Estos comentarios deberían ser lo suficientemente ilustrativos como para permitir, meses después, en caso de necesitar recuperar algo *del pasado*, entender qué tipo de cambios se llevaron a cabo en cada revisión.

#### 5.5.7. Abreviaturas de comandos y opciones

Podemos aprovechar aquí para introducir el uso de las abreviaturas de los comandos y las opciones. Por ejemplo, la línea anterior puede introducirse también como:

```
$ svn ci -m "Comentarios sobre los cambios realizados"
```

#### 5.5.8. Ayuda y documentación de subversion

Las abreviaturas comentadas en el apartado anterior, junto a otras muchas cosas, aparecen en la ayuda del programa:

```
$ svn --help
```

Y si se necesita más información de algún comando concreto, como por ejemplo del comando `commit`, es posible obtenerla introduciendo

```
$ svn --help commit
```

En cualquier caso, todo lo que se necesita saber de *subversion*, y mucho más, lo podremos encontrar en la documentación oficial, recogida en [8].

#### 5.5.9. Concepto de revisión

Con cada acción `commit` creamos, por así decirlo, un nuevo estado del proyecto en el repositorio. Este *estado* recibe el nombre de *revisión*. Así, con nuestro primer `svn commit` crearemos la revisión r1. Con el siguiente, la revisión r2, y así sucesivamente. Al ejecutar el comando, *subversion* devolverá, junto a otra información de interés, el número de la revisión que acaba de ser enviada al repositorio, siempre identificada por un número entero. Del estado de todos los elementos del proyecto en una revisión dada, podremos en cualquier momento recuperar o consultar todo aquello que se necesite, tal y como se verá más adelante.

#### 5.5.9.1. Revisiones Head y Base

Más adelante será de utilidad saber qué se entiende por revisiones *Head* y *Base*. Veamos brevemente de qué se trata.

- La revisión *Head* es, simplemente, la revisión más reciente que se encuentra en el repositorio (es decir, la última revisión enviada por alguno de los usuarios).
- Cuando se genera la copia de trabajo, y también cuando ésta se actualiza al realizar un `commit` o un `update`, *subversion* almacena una copia oculta de todos los archivos dentro de la propia carpeta de trabajo. Ésta última revisión almacenada en la copia de trabajo (oculta para el usuario) es la conocida como revisión *Base*. Dicho de otro modo, la revisión *Base* era la revisión *Head* en el momento del `commit` o del `update` (siempre que no hubieran conflictos), pero no tiene porqué seguir siéndolo.

#### 5.5.10. Actualización de contenidos

Después de haber generado una copia de trabajo en un lugar o puesto de trabajo determinados, es siempre posible que el contenido del repositorio haya sufrido cambios realizados por algún colaborador, o incluso por uno mismo desde otro puesto de trabajo. Para actualizar, en cualquier momento, la copia de trabajo, se realizará la acción

```
$ svn update
```

#### 5.5.11. Conocimiento del estado actual de archivos y carpetas

Siempre es útil conocer el estado de los archivos y carpetas existentes en la copia de trabajo. Es decir, saber, por ejemplo, qué archivos están o no bajo control de versiones, o cuáles han sido modificados desde el último 'commit' al repositorio.

Las figuras 5.4 y 5.5 muestran la manera en que *TortoiseSVN* y *RapidSVN* informan del estado de cada elemento. En ambos casos, el símbolo de interrogación indica que el elemento no está bajo control de versiones, y el rojo que ha sido modificado desde la última actualización del repositorio.

En *subversion*, el comando para conocer el estado de la copia de trabajo es

```
$ svn status
```

del que una posible salida es

## Control de versiones con subversion

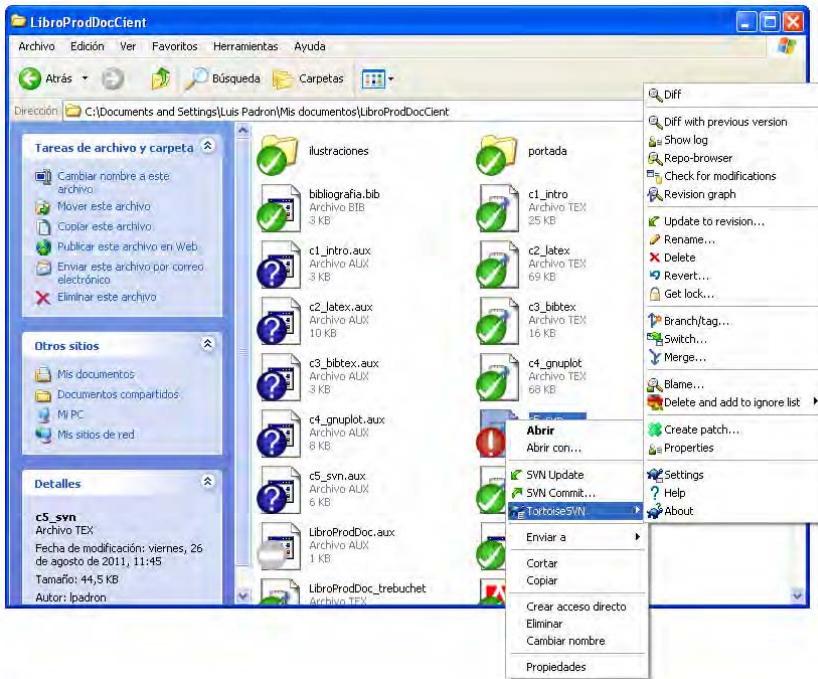


Figura 5.4: Vista del estado actual de una copia de trabajo en TortoiseSVN

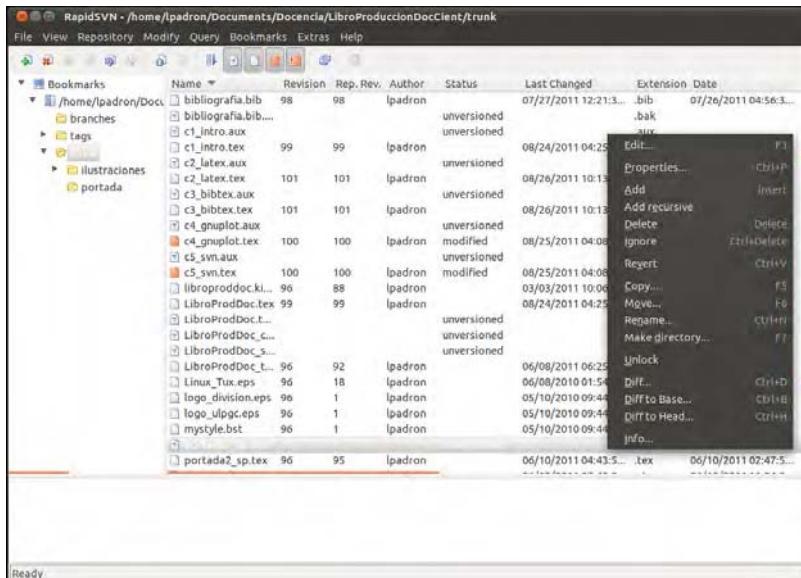


Figura 5.5: Vista del estado actual de una copia de trabajo en RapidSVN

```
?      ilustraciones/CapSVN/status-tortoisesvn.eps
M      c5-svn.tex
```

indicando que el archivo `status-tortoisesvn.eps`, situado dentro de la carpeta `CapSVN` de la carpeta `ilustraciones` no está bajo control de versiones, y que el archivo `c5-svn.tex` ha sido modificado. En este caso, se debería poner la imagen bajo supervisión de *subversion* con el comando `svn add` y, posteriormente, enviar los cambios al repositorio con un `svn commit`.

La información comentada en el párrafo anterior acerca del estado de archivos y carpetas se extrae de los símbolos a la izquierda del nombre y ruta del archivo. Los símbolos más comunes se muestran en la tabla 5.1.

A	(Added)	Añadido después del último <code>commit</code> o <code>update</code>
D	(Deleted)	Borrado después del último <code>commit</code> o <code>update</code>
I	(Ignored)	Ignorado
M	(Modified)	Modificado desde el último <code>commit</code> o <code>update</code>
R	(Replaced)	Reemplazado después de el último <code>commit</code> o <code>update</code>
?	(Unsupervised)	No supervisado por <i>subversion</i>
!	(Missing or incomplete)	Incompleto o inexistente (borrado sin usar <i>subversion</i> )

**Tabla 5.1: Símbolos más comunes en relación con la consulta del estado de una copia de trabajo**

### 5.5.12. Conocimiento del estado previo de archivos y carpetas

En cualquier momento puede interesar recuperar o consultar todo o parte del trabajo tal y como se encontraba en algún momento anterior. Esto puede llevarse a cabo de diversas maneras:

#### 5.5.12.1. Obteniendo un listado de revisiones

En primer lugar, es interesante consultar la lista de las revisiones que han sido creadas junto con los comentarios con los que se documentaron cada una de ellas, lo cual se realiza con el comando

```
$ svn log -v
```

que proporcionará una lista con tantas entradas como revisiones y con los siguientes datos para cada una de ellas: nº de revisión, autor de la revisión,

```

A: ie/lpadron/Documents/Docencia/LibroProduccionDocClient/trunk/c5_svn.tex
B: /tmp/c5_svn-80.tex

```

Top line 132 Encoding: UTF-8 Line end style: Unix

```

% ocasiones, un mayor catálogo de opciones, si bien el uso diario pue
% Para empezar, consideraremos que vamos a guardar nuestro trabajo lo
% comentaremos otras posibilidades.

\begin{figure}[t]
\begin{center}
\includegraphics[width=\textwidth,keepaspectratio]{ilustraciones/Cap5/figuras/ciclo_trabajo_en_uso_sistema_control_versiones.png}
\end{center}
\end{figure}

\subsubsection{Creación de un repositorio}\label{CreacionRepositorio}

El primer paso es construir el almacén central donde \texttt{\$svnadmin create} se desee poner bajo control de versiones. Este almacén recibe el nombre del comando \verb!$svnadmin create!\footnote{El usuario nunca trabajará directamente en el repositorio, sino en sus directorios habrá de comunicarse con el repositorio y realizar las tareas oportunas. De hecho, que se ha generado una serie de archivos y directorios opacos para ello, \texttt{\$textbf{f}nunca} deberá modificar directamente el contenido de un directorio \texttt{\$repository/footnote} (Note que la barra a utilizar para denotar los directorios es siempre /, incluso cuando se trabaja en el mismo directorio).} Para empezar, consideraremos que el trabajo que ha generado una serie de archivos y directorios opacos para el usuario. Más adelante se comentarán otras posibilidades. De este modo, \texttt{\$textbf{f}nunca} deberá modificar directamente el contenido de un directorio \texttt{\$repository/footnote} (Note que la barra a utilizar para denotar los directorios es siempre /, incluso cuando se trabaja en el mismo directorio).
```

```

\begin{verbatim}
$ svnadmin create MiRepo
$ svnadmin Create '/home/lpadron/MiRepo'
$ svnadmin Create "C:/Documents and Settings/Luis Padron/
                  Mis documentos/MiRepo"
\end{verbatim}

Para empezar, consideraremos que el trabajo va a ser almacenado localmente.

```

Top line 132 Encoding: UTF-8 Line end style: Unix

```

% ocasiones, un mayor catálogo de opciones, si bien el uso diario pue
% Para empezar, consideraremos que vamos a guardar nuestro trabajo lo
% comentaremos otras posibilidades.

\subsubsection{Creación de un repositorio}\label{CreacionRepositorio}

El primer paso es construir el almacén central donde \texttt{\$svnadmin create} se desee poner bajo control de versiones. Este almacén recibe el nombre del comando \verb!$svnadmin create!\footnote{El usuario nunca trabajará directamente en el repositorio, sino en sus directorios habrá de comunicarse con el repositorio y realizar las tareas oportunas. De hecho, que ha generado una serie de archivos y directorios opacos para el usuario. Más adelante se comentarán otras posibilidades. De este modo, \texttt{\$textbf{f}nunca} deberá modificar directamente el contenido de un directorio \texttt{\$repository/footnote} (Note que la barra a utilizar para denotar los directorios es siempre /, incluso cuando se trabaja en el mismo directorio).} Para empezar, consideraremos que el trabajo que ha generado una serie de archivos y directorios opacos para el usuario. Más adelante se comentarán otras posibilidades. De este modo, \texttt{\$textbf{f}nunca} deberá modificar directamente el contenido de un directorio \texttt{\$repository/footnote} (Note que la barra a utilizar para denotar los directorios es siempre /, incluso cuando se trabaja en el mismo directorio).
```

```

\begin{verbatim}
$ svnadmin create MiRepo
$ svnadmin Create '/home/lpadron/MiRepo'
$ svnadmin Create "C:/Documents and Settings/Luis Padron/
                  Mis documentos/MiRepo"
\end{verbatim}

\subsubsection{Sobre la estructura recomendada de un repositorio: tags y

```

Figura 5.6: Ejemplo de diferenciación entre archivos con *KDiff3*

fecha de la revisión, cambios que tuvieron lugar, y comentarios asociados a la revisión.

#### 5.5.12.2. Obteniendo diferencias entre distintos instantes

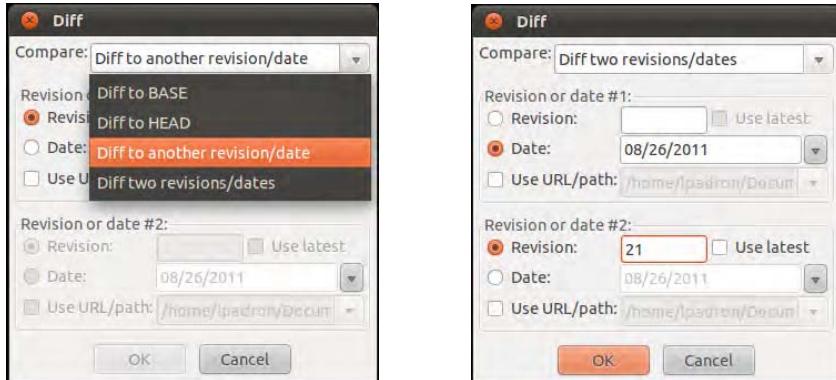
Por otro lado, podemos querer conocer las diferencias entre el estado de los archivos en la revisión *Base* (ver 5.5.9.1) y en el momento actual en la copia de trabajo. Desde la línea de comandos, se introduciría

```
$ svn diff
```

obteniendo una descripción de las diferencias en un formato denominado *unified diff format* (relativamente sencillo, pero sobre el que no entraremos en detalles aquí para no extender demasiado el capítulo). Si, en lugar de hacerlo desde la línea de comandos, se hace desde cualquiera de las interfaces gráficas haciendo *clic* con el botón derecho sobre el archivo en cuestión, y eligiendo la opción *diff*, se obtendría una presentación mucho más gráfica de las diferencias, tal y como se representa en la figura 5.6, donde, a la izquierda puede verse este archivo en el momento de la redacción, y a la derecha una revisión anterior del mismo archivo.

Tal y como puede consultarse en la ayuda del programa, las posibilidades del comando *diff* son mucho mayores, y permite comparar cualesquier dos estan-

dos de un archivo determinado, entre otras cosas. Dado que esto pretende ser sólo una introducción a *subversion*, no entraremos más en detalle. Sin embargo, la figura 5.7 da una idea de las posibilidades a través del diálogo *diff* de *RapidSVN*.



**Figura 5.7: Ejemplos del diálogo de *RapidSVN* para el comando *diff***

### 5.5.12.3. Recuperando un archivo de una revisión anterior

Obviamente, también puede interesar consultar el contenido de un archivo determinado en una revisión dada, lo que puede hacerse mediante el comando *cat*. Para listar el contenido del archivo 'indice.tex' en la revisión 3, se introduciría:

```
$ svn cat -r 3 indice.tex
```

Puede guardarse una copia de dicho archivo con otro nombre (por ejemplo 'indice.tex.r3') de la siguiente manera:

```
$ svn cat -r 3 indice.tex > indice.tex.r3
```

### 5.5.12.4. Recuperando un archivo de la revisión actual

O quizás, después de haber modificado un archivo, se decide, por una razón u otra, recuperar el archivo tal y como estaba en la revisión Base (tras el último *commit* o *update*). Para ello, se utilizaría el comando *revert*:

```
$ svn revert indice.tex
```

### 5.5.12.5. Deshacer cambios enviados anteriormente

Otra situación que podemos encontrarnos en el día a día es la necesidad de rehacer todos los cambios introducidos en una revisión dada. Imagínese, por ejemplo, que en un momento dado se eliminan o modifican algunos elementos (líneas de texto, archivos ...) y se ejecuta posteriormente un `svn commit` que da lugar a la revisión 48. Un tiempo después, cuando ya se han enviado muchos más cambios y se va por la revisión 116, se desea recuperar aquellos elementos que nunca debieron haber sido modificados o eliminados. Cualquiera de los siguientes comandos (totalmente equivalentes entre sí) revocaría dichos cambios sin afectar al resto de cambios en revisiones anteriores o posteriores:

```
$ svn merge --change -48 file:///home/lpadron/MiRepo/trunk
$ svn merge -c -48 file:///home/lpadron/MiRepo/trunk
$ svn merge --revision 48:47 file:///home/lpadron/MiRepo/trunk
$ svn merge -r 48:47 file:///home/lpadron/MiRepo/trunk
```

Nótese que hay dos opciones posibles en este caso: la opción `--change` (`-c`) con la que se revoca un cambio concreto (el signo menos delante del número de la revisión es lo que indica que se desea revocar los cambios); y la opción `--revision` (`-r`), que se continúa con el rango de revisiones que se necesita revocar en sentido inverso (con esta opción, se puede revocar un conjunto consecutivo de revisiones en lugar de sólo una).

### 5.5.13. Conflictos

Imaginemos la siguiente situación. Dos colaboradores de un mismo proyecto han estado trabajando sobre un mismo fichero, y dentro de él, han modificado partes comunes de su contenido, posiblemente sin saberlo. Ambos parten de una misma revisión, pero uno de ellos decide, en un momento dado, enviar sus cambios al repositorio a través de un `svn commit`. Los cambios son admitidos en el repositorio sin problema alguno, y una nueva revisión del proyecto es generada. Cuando el segundo colaborador quiera enviar sus cambios, `subversion` detectará que algunos de los cambios que quiere introducir no tienen en cuenta los que ya envió anteriormente su compañero. En este caso, `subversion` notificará la existencia de un *conflicto* y dará distintas opciones al usuario para resolverlo. Estas opciones son:

- **Posponer** la resolución del conflicto, de manera que se permite actualizar el resto de contenidos y se guardan las distintas versiones en conflictos para ser estudiadas posteriormente por el usuario.
- **Mostrar las diferencias** entre los ficheros en conflicto.

- **Editar el fichero en conflicto.**
- **Resolver** el conflicto después de haber editado el fichero bajo la responsabilidad del usuario.
- **Aceptar mi versión** del archivo en conflicto, lo que implica desechar completamente los cambios propuestos por el otro colaborador.
- **Desechar mi versión** del archivo en conflicto, lo que implica aceptar todos los cambios propuestos por el otro colaborador y perder los propuestos por uno mismo.

Explicado el concepto, y siendo su uso relativamente sencillo e intuitivo, vamos a evitar aquí entrar en más detalles, pero para más información, el lector interesado puede consultar, por ejemplo, la descripción dada en [8].

#### 5.5.14. Creación de tags y branches

En la sección 5.5.2 se comentó la estructura recomendada de los proyectos, con una carpeta *trunk* donde se desarrollaría el trabajo del día a día, una carpeta *tags* donde conservar estados importantes del trabajo, y una carpeta *branches* donde, en caso necesario, desarrollar variaciones del proyecto. Recordemos también que esta estructura no es, en absoluto, obligatoria, ya que dichas carpetas no son más que eso, carpetas como cualquier otra, y por tanto pueden estar o no, o pueden tener nombres diferentes sin ningún tipo de problema.

Imaginemos que se desea conservar e identificar el estado del proyecto tal y como está en un momento dado (por ejemplo, el primer borrador del libro), conservando a su vez toda su historia y haciéndolo disponible desde el repositorio. Se trataría, simplemente, de copiar el contenido del proyecto a otra carpeta, digamos *tags/borrador-1/*, con lo que se estaría creando un tag. Por otro lado, si lo que se quiere es crear una variante, también se debería realizar una copia del contenido del proyecto, digamos que a *branches/variante-1/*.

Por tanto, ambas cosas se reducen a la copia de un directorio en otro, lo cual se realizaría con el comando *svn copy* aplicado directamente sobre el repositorio. Para el segundo caso quedaría, por ejemplo, como:

```
$ svn copy file:///home/lpadron/MiRepo/trunk  
file:///home/lpadron/MiRepo/branches/variante-1/  
-m "creando una variante en la carpeta branches para  
hacer una variante de estos apuntes para otro curso"
```

donde ambas direcciones son la URL de la carpeta deseada dentro del repositorio en cuestión, sea cual sea su localización.

Utilizando estos comandos se crea dentro del repositorio una nueva carpeta con una copia del proyecto. Si se quiere posteriormente trabajar en esa variante, se deberá obtener en algún lugar del ordenador una copia de trabajo de esta variante, utilizando el comando `checkout` tal y como se vio en la sección 5.5.3. Nótese que es posible tener una copia de trabajo para la línea principal del proyecto, y otras para las variantes del mismo, lo que permite trabajar en ambas simultáneamente y enviar los cambios al repositorio de manera independiente y sin que los cambios en uno afecten a los otros.

Sin embargo, siempre es posible que haya cambios que se deseen realizar tanto en la línea principal como en las variantes (por ejemplo, corrección de faltas de ortografía o erratas en las zonas comunes), y también es posible que en algún momento se quiera volver a reunificar ambas líneas de desarrollo. Esto se realiza mediante el comando

```
$ svn merge
```

capaz de controlar los cambios que se realizan en cada lugar y de unificarlos de manera controlada. De entre las muchas posibilidades que ofrece *subversion* en este sentido (véase [8] para más información), imaginemos a modo de ilustración que se quieren incorporar los cambios realizados en *trunk* en la revisión 224 a la *variante-1*. Para ello, el comando a usar podría ser:

```
$ svn merge -c 224 file:///home/lpadron/MiRepo/trunk
```

Por otro lado, si se quieren introducir los cambios hechos en un conjunto de revisiones consecutivas, por ejemplo, de la 224 a la 254, el comando sería

```
$ svn merge -r 224:254 file:///home/lpadron/MiRepo/trunk
```

## 5.6. Repositorios en red

Cuando se creó el repositorio en la sección 5.5.1 y posteriormente la copia local de trabajo en la sección 5.5.3, se dio por sentado que el repositorio sería local, es decir, que estaría situado en el PC del usuario, el mismo PC en el que se trabajaría. Si usted trabaja sólo, utiliza normalmente un único ordenador, y no necesita acceder al repositorio desde otro PC distinto, entonces un repositorio local es exactamente lo que usted necesita.

Sin embargo, algo más hará falta en el caso de un proyecto en el que vayan a participar varias personas, o en el que una única persona necesite acceder al repositorio desde distintos puestos de trabajo. En tal caso, el repositorio deberá ser accesible en red, ya sea en una red interna o desde Internet.

Hacerlo no es realmente difícil, y además puede realizarse de distintas maneras, lo que permitirá elegir la opción más adecuada en cada caso. Entre

otras posibilidades, puede crearse el repositorio en una máquina a la que se tenga acceso a través del protocolo *SSH*, puede utilizarse la herramienta *svnserve*, o puede utilizarse un servidor Apache<sup>11</sup>. En estos casos, el repositorio ha de ser creado en una máquina que funcione como *servidor de Internet*. Para ello, una posibilidad obvia es poseer y configurar nuestro propio servidor, pero otra posibilidad es hacer uso de servicios de *hosting* en Internet. Este tipo de servicios, algunos gratuitos y otros de pago, son ofrecidos por empresas e instituciones y ofrecen la posibilidad de disponer de repositorios con acceso desde internet (una lista puede consultarse, por ejemplo, en <http://www.svnhostingcomparison.com>). Algunos de ellos ofrecen este servicio especialmente para el desarrollo de programas de software libre, mientras otros son de uso completamente general.

En cualquier caso, todas las operaciones y conceptos vistos en este capítulo son independientes de que el repositorio sea local o remoto. El único cambio sustancial reside en la manera de escribir la dirección del repositorio. Por ejemplo, si para obtener una copia de trabajo de un repositorio local se escribía

```
$ svn checkout file:///home/lpadron/MiRepo MiCopiaDeTrabajo
```

para obtenerla de un repositorio en otra máquina a la que se accede por *SSH* se introduciría, por ejemplo:

```
$ svn checkout  
    svn+ssh://direccion.es/home/lpadron/MiRepo MiCopiaDeTrabajo
```

Sin embargo, el resto de las operaciones se realizarían tal y como se ha visto hasta ahora.

---

<sup>11</sup> Nótese que cada una de estas opciones conlleva su propias especificidades, y su exposición detallada escapa al campo que pretende abarcar este manual. Una vez que usted conoce las posibilidades, y en función de su situación específica, se anima una vez más al lector interesado a consultar [8] o la documentación propia del servicio utilizado.

## Bibliografía

- [1] Lamport, L. (1985) *L<sup>A</sup>T<sub>E</sub>X- A Document Preparation System*. Addison-Wesley.
- [2] Knuth, D. E. (1986) *The T<sub>E</sub>Xbook*. Addison-Wesley.
- [3] Mittelbach, F. and Goossens, M. (2004) *The L<sup>A</sup>T<sub>E</sub>X Companion (Second Edition)*. Addison-Wesley.
- [4] Engelen, J. B. C. (2010) *How to include an SVG image in L<sup>A</sup>T<sub>E</sub>X*. Disponible para su descarga en <http://ftp.udc.es/CTAN/info/svg-inkscape/InkscapePDFLaTeX.pdf>.
- [5] Kopka, H. and Daly, P. W. (2004) *Guide to L<sup>A</sup>T<sub>E</sub>X*. Addison-Wesley, Pearson Education.
- [6] Cascales Salinas, B., Lucas Saorín, P., Mira Ros, J. M., Pallarés Ruiz, A. J., and Sánchez-Pedreño Guillén, S. (2003) *El libro de L<sup>A</sup>T<sub>E</sub>X*. Prentice Hall, Pearson Educación.
- [7] Janert, P. K. (2010) *Gnuplot in action*. Manning Publications Co.
- [8] Collins-Sussman, B., Fitzpatrick, B. W., and Pilato, C. M. (2008) *Version control with subversion*. Disponible para su descarga en <http://svnbook.red-bean.com/>.