



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**Ingeniería Técnica Informática de Sistemas**

**Clase  $\LaTeX$  para Proyecto Fin de Carrera**

**Realizado por  
Fernando Antonio Caro Vega  
Francisco Javier Delgado Villegas**

**Dirigido por  
José Ramón Portillo**

**Departamento  
Matemática Aplicada I**

**Sevilla, (09/2008)**

---

## Resumen

---

Esta documentación corresponde a un proyecto de final de carrera consistente en la creación de una clase  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  llamada `pclass`, dicha clase tiene como finalidad el formateo de memorias de proyectos pertenecientes a la Escuela Técnica Superior de Ingeniería Informática. A pesar de ello con algunas modificaciones no resultará complicado adaptar esta plantilla para aplicarla al resto de titulaciones.

A lo largo de esta memoria se detallará el procedimiento seguido para la creación partiendo de cero de `pclass`. Además de este proceso de creación, también se contemplarán las distintas funcionalidades propias de la clase, las cuales facilitarán de manera más que notoria la redacción de una memoria.

Finalmente si lo que te interesa es pasar directamente a redactar el contenido de tu memoria, puedes omitir los distintos pasos seguidos para crear `pclass` y dirigirte al Capítulo 11. En este capítulo puedes encontrar un manual completo de uso de la clase que da origen a este proyecto.



---

## Agradecimientos

---

Resulta muy complicado para los autores de este proyecto enumerar a todos aquellos que han contribuido al buen fin del mismo. Por este motivo queremos disculparnos si alguien que ha resultado ser un apoyo a lo largo del proceso no aparece explícitamente en estas líneas.

Sobre todo agradecer a nuestras familias la paciencia y la comprensión sin la cual no hubiera sido posible superar aquellas situaciones adversas que nos han surgido. Y por supuesto a todos esos amigos, compañeros de trabajo, etc., que han soportado nuestras charlas acerca de algo tan desconocido para ellos como el lenguaje  $\text{\LaTeX}$ , muchas gracias a todos.



---

## Índice general

---

Índice general	V
Índice de cuadros	IX
Índice de figuras	XI
Índice de código	XIII
<b>1 Definición de objetivos</b>	<b>1</b>
<b>2 Introducción</b>	<b>5</b>
2.1 ¿Qué es T <sub>E</sub> X?	5
2.1.1 ¿Qué es L <sup>A</sup> T <sub>E</sub> X?	5
2.1.2 ¿Qué es L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> ?	6
2.2 L <sup>A</sup> T <sub>E</sub> X vs WYSIWYG	7
2.2.1 Ventajas e inconvenientes de L <sup>A</sup> T <sub>E</sub> X	7
<b>3 Conceptos básicos</b>	<b>9</b>
3.1 Primeros pasos en L <sup>A</sup> T <sub>E</sub> X	9
3.1.1 Instalación de L <sup>A</sup> T <sub>E</sub> X y T <sub>E</sub> X	10
3.2 Lo que necesita saber sobre L <sup>A</sup> T <sub>E</sub> X	12
3.2.1 Ficheros de entrada de L <sup>A</sup> T <sub>E</sub> X	12
3.2.2 Estructura de un fichero de entrada	14
3.2.3 El formato del documento	15
3.2.4 Proyectos grandes	19
3.3 ¿Qué documentación hay sobre T <sub>E</sub> X/L <sup>A</sup> T <sub>E</sub> X?	20
3.3.1 Tutoriales	20
3.3.2 Libros	20
3.3.3 Varios	22
<b>4 Análisis de requisitos, diseño e implementación</b>	<b>23</b>
4.1 Diseño e implementación	23
4.2 Análisis de requisitos	25

<b>5</b>	<b>Análisis de antecedentes y aportación realizada</b>	<b>35</b>
<b>6</b>	<b>Análisis temporal y costes de desarrollo</b>	<b>37</b>
6.1	Análisis temporal . . . . .	37
6.2	Costes de desarrollo . . . . .	39
6.2.1	Inversiones . . . . .	39
6.2.2	Costes de software . . . . .	39
6.2.3	Costes de hardware . . . . .	40
6.2.4	Costes indirectos . . . . .	40
6.2.5	Costes de personal . . . . .	41
6.2.6	Presupuesto . . . . .	41
<b>7</b>	<b>Comparación con otras alternativas</b>	<b>43</b>
<b>8</b>	<b>Pruebas y conclusiones</b>	<b>47</b>
8.1	Pruebas . . . . .	47
8.2	Conclusiones . . . . .	47
<b>9</b>	<b>Clase proyecto fin de carrera pclass</b>	<b>49</b>
9.1	Introducción . . . . .	49
9.2	Estructura de la clase . . . . .	49
9.2.1	Identificación . . . . .	50
9.2.2	Usando una clase . . . . .	50
9.2.3	Declarando opciones . . . . .	51
9.2.4	Clase mínima de ejemplo . . . . .	53
9.2.5	Código de opciones clase <code>pclass</code> . . . . .	54
9.3	Clase <code>pclass</code> . . . . .	54
9.3.1	Frontmatter . . . . .	56
9.3.2	Mainmatter . . . . .	58
9.3.3	Backmatter . . . . .	60
9.4	Paquetes . . . . .	60
9.4.1	¿Cómo y dónde instalo nuevos paquetes? . . . . .	60
9.4.2	<code>fancyhdr</code> . . . . .	65
9.4.3	<code>titlesec</code> . . . . .	75
9.4.4	<code>tocbibind</code> . . . . .	80
9.4.5	<code>listings</code> . . . . .	82
9.5	Bibliografía . . . . .	85
9.6	Macros . . . . .	99
9.6.1	Macros de la clase . . . . .	101
9.7	Entornos . . . . .	104
9.8	Ortotipografía . . . . .	105
9.8.1	Ortotipografía de programas informáticos . . . . .	106

---

<b>10 Errores</b>	<b>109</b>
10.1 Errores de compilación . . . . .	109
10.2 Correcciones . . . . .	111
<b>11 Manual</b>	<b>113</b>
11.1 Manual de usuario . . . . .	113
11.2 Introducción . . . . .	113
11.3 ¿Cómo consigo L <sup>A</sup> T <sub>E</sub> X?	114
11.3.1 Para Unix, Linux, etc. . . . .	114
11.3.2 Para Windows . . . . .	114
11.4 Mi primer documento . . . . .	115
11.4.1 Usando T <sub>E</sub> XnicCenter . . . . .	115
11.4.2 Usando consolas o terminales . . . . .	116
11.4.3 Sugerencias . . . . .	117
11.5 Formato de la memoria de tu proyecto . . . . .	117
11.5.1 Datos de tu proyecto . . . . .	118
11.5.2 Capítulos . . . . .	119
11.6 Notación matemática . . . . .	120
11.6.1 Construcciones básicas . . . . .	121
11.6.2 Cuadros de símbolos . . . . .	123
11.7 Editando en L <sup>A</sup> T <sub>E</sub> X . . . . .	124
11.7.1 Secciones . . . . .	125
11.7.2 Entornos . . . . .	125
11.7.3 Texto enfatizado . . . . .	126
11.8 Bibliografía . . . . .	127
11.8.1 Archivo de biblioteca virtual . . . . .	128
11.8.2 Citas bibliográficas . . . . .	129
11.9 Algunos conceptos importantes . . . . .	130
11.9.1 Cuadros . . . . .	131
11.9.2 Figuras . . . . .	132
11.9.3 Conceptos matemáticos útiles . . . . .	134
11.9.4 Alguna información de interés . . . . .	135
11.10 Información de la clase pclass.cls . . . . .	136
11.11 Archivos adjuntos a pclass.cls . . . . .	137
<b>Licencia</b>	<b>139</b>
<b>Bibliografía</b>	<b>147</b>



---

## Índice de cuadros

---

3.1	Clases de documentos . . . . .	16
3.2	Opciones de clases de documentos . . . . .	17
3.3	Algunos paquetes distribuidos con L <sup>A</sup> T <sub>E</sub> X . . . . .	18
3.4	Estilos de página predefinidos en L <sup>A</sup> T <sub>E</sub> X . . . . .	19
6.1	Planificación del proyecto . . . . .	39
6.2	Presupuesto del proyecto . . . . .	41
9.1	Opciones para book.cls . . . . .	55
9.2	Jerarquía de las unidades de estructura según la clase book . . . . .	55
9.3	Selectores . . . . .	69
9.4	Lenguajes predefinidos para paquete listings . . . . .	83
11.1	Entornos basados en amsthm . . . . .	127
11.2	Ventas empresa vinícola . . . . .	131



---

## Índice de figuras

---

2.1	Donald E.Knuth . . . . .	6
3.1	Emacs . . . . .	11
3.2	Kile . . . . .	12
3.3	Fichero mínimo de $\text{\LaTeX}$ . . . . .	15
3.4	Ejemplo de artículo científico en español . . . . .	15
4.1	Formato definido en pclass para la portada . . . . .	26
7.1	Etapas de creación de un documento $\text{\LaTeX}$ . . . . .	45
9.1	Diseño de página fancyhdr . . . . .	69
9.2	Ejemplo a una cara fancyhdr . . . . .	70
9.3	Ejemplo a dos caras fancyhdr . . . . .	71
9.4	Diseño por defecto página par fancyhdr . . . . .	71
9.5	Diseño por defecto página impar fancyhdr . . . . .	72
9.6	Marcadores de capítulos . . . . .	73
9.7	Diseño de clase para páginas impares . . . . .	74
9.8	Diseño de clase para páginas pares . . . . .	74
9.9	Mínimo ejemplo de listings . . . . .	82
9.10	Ejemplo lstset de paquete listings . . . . .	83
9.11	Nombres en bibliografía . . . . .	86
9.12	Nombre y título en bibliografía . . . . .	87
9.13	Editorial y año en bibliografía . . . . .	87
9.14	Vista general interfaz gráfica de JabRef . . . . .	89
9.15	Vista de detalle interfaz gráfica de JabRef . . . . .	90



---

## Índice de código

---

4.1	Datos para completar la portada . . . . .	27
4.2	Macro para hacer la portada . . . . .	27
4.3	Contenido de frontmatter . . . . .	28
4.4	Macros cuadro y figura . . . . .	29
4.5	Comandos para generar el índice de código . . . . .	30
4.6	Definiciones relativas al paquete listings . . . . .	31
4.7	Apéndices . . . . .	32
4.8	Comandos para incluir la bibliografía . . . . .	33
9.1	Código de opciones de clase pclass . . . . .	54
9.2	Frontmatter en proyect.tex . . . . .	56
9.3	Modificación de tabla y código . . . . .	57
9.4	Nivel de profundidad . . . . .	57
9.5	Creación de portada . . . . .	57
9.6	Mainmatter en proyect.tex . . . . .	58
9.7	ExecuteOptions en pclass.cls . . . . .	58
9.8	Páginas sin información en blanco . . . . .	58
9.9	Dimensión de páginas . . . . .	59
9.10	Redefinición itemize . . . . .	60
9.11	Backmatter . . . . .	60
9.12	Código de Clase para diseño de cabeceras y pies de página . . . . .	74
9.13	Listings . . . . .	84
9.14	Macros sobre L <sup>A</sup> T <sub>E</sub> X . . . . .	101
9.15	Macro para código fuente . . . . .	101
9.16	Macro para insertar imágenes . . . . .	102
9.17	Macro para insertar cuadros . . . . .	102
9.18	Macros para variables de portada . . . . .	103
9.19	Entorno para amsthm . . . . .	104
9.20	Entorno para cuadro vacío . . . . .	104
10.1	Correcciones . . . . .	112



# CAPÍTULO 1

---

## Definición de objetivos

---

Para comenzar este capítulo, debemos mencionar que con el desarrollo del proyecto objeto de esta documentación se pretende elaborar una clase  $\text{\LaTeX}$  destinada al formateo de memorias de proyectos. De este modo el usuario de `pclass`, obtendrá como resultado final una memoria acorde al formato definido por la Univesidad de Sevilla para este tipo de documentos.

En lo que respecta a los objetivos de este proyecto de fin de carrera, debemos resaltar el hecho de que dichos objetivos pueden encuadrarse en dos grandes bloques bien diferenciados.

Por un lado nos encontramos con los objetivos implícitos de todo documento escrito usando el lenguaje de programación  $\text{\LaTeX}$ . Dichos objetivos han contribuido de manera determinante a una gran difusión de dicho lenguaje en el ámbito científico, convirtiéndose en el estándar exigido para la publicación de resultados. Dentro de este bloque podríamos enumerar los siguientes:

- Generación de documentos de gran calidad, fundamentalmente cuando aparecen involucrados textos que incluyen numerosas fórmulas matemáticas, ecuaciones, tablas, etc.
- Posibilidad de albergar en un mismo documento  $\text{\LaTeX}$ , en nuestro caso una memoria de un proyecto de fin de carrera, texto ordinario junto con texto escrito en modo matemático.
- Liberar al usuario final de esta clase  $\text{\LaTeX}$  de la necesidad de definir aspectos comunes en todo documento o memoria de un proyecto de fin de carrera como por ejemplo: la clase de documento, indicaciones sobre márgenes, largo y ancho de página, numeración, etc. Esta tarea puede resultar especialmente tediosa cuando tenemos que ajustar nuestro documento a una determinada serie de especificaciones.

En nuestro caso dichas especificaciones obedecerán al reglamento de la Universidad de Sevilla para la asignatura proyecto informático de las titulaciones: Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas.

- Separar en dos campos independientes y bien diferenciados:
  1. Todos los aspectos relacionados con la apariencia de la memoria del proyecto de fin de carrera. Todos estos aspectos serán englobados en un archivo `.cls`, en nuestro caso se tratará del `pclass.cls`, siendo dicha clase el objeto de esta memoria.
  2. La estructura lógica de la memoria del proyecto de fin de carrera. En este apartado es en el que debe centrar su atención el usuario, lo hará editando, a modo de plantilla, el contenido de un archivo `.tex`. En nuestro caso se tratará del `project.tex`, en el que el usuario introducirá el contenido de su memoria.

Además por otra parte, también podemos distinguir las distintas finalidades por las que nos parece muy necesaria la creación de una clase  $\text{\LaTeX}$  para formatear proyectos de fin de carrera. De entre dichas finalidades podríamos destacar las siguientes:

- Conseguir que cualquier persona, sin conocimientos previos acerca del lenguaje  $\text{\LaTeX}$ , sea capaz de asimilar una serie de nociones básicas de dicho lenguaje. Dichas nociones irán encaminadas a que el usuario sea capaz de redactar su memoria de proyecto de fin de carrera de manera sencilla haciendo uso de la clase `pclass.cls`. Evidentemente dicha memoria se regirá por el reglamento establecido por la Universidad de Sevilla para su presentación.
- Definir, a través de la clase  $\text{\LaTeX}$  objeto de esta memoria, todas las especificaciones necesarias según el reglamento de de la Universidad de Sevilla para la asignatura proyecto informático de las titulaciones: Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas. De este modo intentamos conseguir un objetivo primordial, que no es otro que centrar la atención del usuario de la clase en la estructura lógica de la memoria de su proyecto de fin de carrera. Una vez conseguido lo anterior, la apariencia del documento pasará a un segundo plano en lo que respecta al usuario.
- Facilitar lo máximo posible al usuario de la clase `pclass.cls` algunas de las operaciones más comunes a la hora de escribir un documento en  $\text{\LaTeX}$ . Para ello agrupamos, haciendo uso de macros, algunas

sentencias de comandos de uso común. Un ejemplo de macro puede ser `\hacerportada`, la cual nos presentará la portada de nuestro proyecto según la normativa de la Universidad de Sevilla. Para información más detallada acerca de estos aspectos podemos consultar el manual de usuario.



### Introducción

---

#### 2.1– ¿Qué es T<sub>E</sub>X?

El creador de Tex es Donald E. Knuth, su trabajo fue un encargo de la American Mathematical Society a principios de los años 70. Esta sociedad buscaba un lenguaje para formatear sus artículos llenos de teoremas y fórmulas matemáticas de gran complejidad. El resultado obtenido fue un lenguaje extremadamente potente, pero también difícil de aprender y usar.

Basta decir que de hecho el sistema L<sup>A</sup>T<sub>E</sub>X es el estándar de creación de textos científicos desde hace muchos años, sin embargo aprender L<sup>A</sup>T<sub>E</sub>X no es cosa de un día, no es algo fácil pero tampoco imposible, de forma que con algo de paciencia se pueden conseguir resultados casi inmediatos. Para facilitar el trabajo con T<sub>E</sub>X surgieron numerosas macros que agrupaban distintas instrucciones de T<sub>E</sub>X.

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

L<sup>A</sup>T<sub>E</sub>X es un paquete de macros, especialmente diseñado para la creación de textos técnicos y científicos, que permite componer e imprimir documentos de forma sencilla, con la mayor calidad tipográfica, utilizando para ello patrones previamente definidos. Está basado en un lenguaje de composición de bajo nivel llamado T<sub>E</sub>X y facilita el uso de este potente lenguaje.

A diferencia de otros sistemas para procesar textos, no se obtiene el resultado final a medida que se va escribiendo sino que primero se crea un código fuente y seguidamente se procesa para llegar al documento final, en este sentido se asemeja mucho a los lenguajes de marcas como el HTML.

L<sup>A</sup>T<sub>E</sub>X fue escrito por Leslie Lamport en los años 80 y actualmente multitud de libros, revistas científicas están escritas íntegramente en L<sup>A</sup>T<sub>E</sub>X,



Figura 2.1: Donald E.Knuth

incluso en numerosos foros científicos se ha convertido en el estándar exigido para la publicación de resultados. Una de las razones de la gran difusión de  $\text{\LaTeX}$  es su precio.  $\text{\LaTeX}$  es freeware, es decir, puede conseguirse a través de internet y utilizarse de forma gratuita y legal. Sin embargo la ventaja fundamental entre  $\text{\LaTeX}$  y otros procesadores más conocidos (Word Perfect, Microsoft Word) es la calidad de los documentos que genera, fundamentalmente cuando aparecen involucrados textos que incluyen numerosas fórmulas matemáticas, ecuaciones, tablas, etc. Además está disponible para la práctica totalidad de sistemas operativos actuales, incluyendo Windows, Linux, Unix ,etc. Otra de sus ventajas es la existencia de una gran cantidad de paquetes estándares pensados para dotar a los textos de toda la funcionalidad que se precise. Así existen paquetes para incluir gráficos, textos de lenguajes de programación, fórmulas físicas y químicas, diagramas matemáticos, etc. Por todo ello  $\text{\LaTeX}$  ha conocido una gran difusión en el ámbito científico, siendo hoy día el procesador más usado por matemáticos, físicos y gran número de ingenieros.

### 2.1.2. ¿Qué es $\text{\LaTeX} 2_{\epsilon}$

Revisión completa desde la versión  $\text{\LaTeX} 2.09$ , que fue durante muchos años la versión standard de  $\text{\LaTeX}$  hasta la aparición de  $\text{\LaTeX} 2_{\epsilon}$ , uno

de sus propósitos centrales fue la integración dentro de un ambiente único de  $\LaTeX$ . La idea fundamental de  $\LaTeX 2_{\epsilon}$  es que toda futura adición o extensión de  $\LaTeX$  se haga por medio de paquetes individuales, que el usuario puede invocar por medio de la instrucción `\usepackage{...}`. De este modo se intenta poner fin a la proliferación de dialectos incompatibles.

## 2.2– $\LaTeX$ vs WYSIWYG

¿Quién no ha enviado un documento escrito con un procesador de textos clásico a una impresora diferente de la de su ordenador y ha obtenido un resultado desastroso, incluyendo cambio de fuentes, modificación de la paginación, etc.?. Todo esto es historia con  $\LaTeX$ . Digamos que en  $\LaTeX$ , el usuario se concentra en la estructura lógica del documento más que en su apariencia, ya que ésta se define aparte. Ello permite modificar de forma rápida y eficaz la apariencia, sin modificar en absoluto el contenido.

$\LaTeX$  desempeña el papel de diseñador tomando parte en el formato del documento (longitud del renglón, tipo de letra, espacios, ...) para darle luego instrucciones al cajista,  $\TeX$ . El tratamiento del texto es totalmente diferente a procesadores tales como Microsoft Word o Word Perfect en los cuales el autor ve en pantalla lo que exactamente aparecerá luego en la impresora. Esto tiene sus ventajas e inconvenientes como comentaremos más adelante.

Se le dará mayor importancia a la legibilidad y comprensión del texto que al aspecto más o menos agradable que este pueda presentar. Con un sistema WYSIWYG <sup>1</sup> podemos obtener textos estéticamente bonitos pero con una estructura muy pequeña o inconsistente. Sin embargo con  $\LaTeX$  esto no está permitido ya que el autor está forzado a seguir un orden e indicar una estructura.

### 2.2.1. Ventajas e inconvenientes de $\LaTeX$

Ventajas:

- Facilita la composición de fórmulas con un cuidado especial.
- Existe mayor cantidad de diseños de textos profesionales a disposición, con lo que realmente se pueden crear documentos como si fueran de imprenta.
- No hace falta preocuparse por los detalles. Sólo es necesario introducir instrucciones para indicar la estructura del documento.
- Las estructuras, tales como notas al pie de página, bibliografía, índices, tablas y muchas otras, pueden ser introducidas sin demasiado esfuerzo.

---

<sup>1</sup>Siglas que significan, What you see is what you get, lo que ve es lo que obtendrá.

- Existen paquetes adicionales, sin coste alguno, para muchas tareas tipográficas aunque no se facilitan directamente por el  $\text{\LaTeX}$  básico.
- $\text{\LaTeX}$  hace que los autores tiendan a escribir textos bien estructurados porque así es como trabaja  $\text{\LaTeX}$ , o sea, indicando su estructura.
- $\text{\TeX}$  es altamente portable y gratis. Por eso, el sistema funciona prácticamente en cualquier plataforma.

Inconvenientes:

- Si bien se pueden ajustar algunos parámetros de un diseño de documento predefinido, la creación de un diseño entero es difícil y lleva mucho tiempo.
- El periodo de aprendizaje es mayor que los WYSIWYG.
- No sirve para maquetación de publicaciones. Se necesita invertir demasiado tiempo.

## CAPÍTULO 3

---

### Conceptos básicos

---

#### 3.1– Primeros pasos en L<sup>A</sup>T<sub>E</sub>X

Desde el punto de vista del usuario, L<sup>A</sup>T<sub>E</sub>X se presenta como un programa de línea de comandos que toma como parámetro principal el fichero fuente que contiene la descripción (texto y comandos L<sup>A</sup>T<sub>E</sub>X) del documento a generar. Existen dos comandos para ejecutar L<sup>A</sup>T<sub>E</sub>X:

1. `latex` genera el documento final en formato DVI (DeVice Independent), a partir del cual puede obtenerse, mediante la aplicación `dvips`, el documento en formato PS (PostScript):

`latex fichero.tex` (genera fichero.dvi, necesita un visualizador específico, pero por la naturaleza del formato, la lectura resulta lenta)

`dvips fichero.dvi -o fichero.ps` (genera fichero.ps, más manejable que DVI y directamente entendible por muchas impresoras láser, se puede ver con GhostScript/GhostView)

2. `pdflatex` genera el documento directamente en formato PDF (Portable Document File, de uso muy extendido en Internet):

`pdflatex fichero.tex` (genera fichero.pdf, visualizable con Adobe Acrobat Reader)

A la hora de incluir gráficos o imágenes en los documentos, hay que tener en cuenta que cada una de estas aplicaciones es capaz de comprender sólo unos ciertos formatos gráficos, si necesitásemos insertar figuras almacenadas en otros formatos no directamente soportados, tendremos que recurrir a un programa que haga la conversión. Por una parte `latex` trabaja cómodamente tan sólo con EPS (Encapsulated PostScript, una variante especial de PS), sin embargo `pdflatex` espera que las figuras estén en PDF (preferible para los gráficos vectoriales), PNG (adecuado para las

capturas de pantalla o cualquier imagen generada por ordenador) o JPG (adecuada para fotografías).

Una vez conocemos todo lo mencionado en el párrafo anterior, ¿cómo hago uso de esos comandos en mi sistema operativo?  $\LaTeX$  puede usarse en Linux (y otros sistemas tipo UNIX) y en MS Windows (aunque parezca sorprendente). Dependiendo del sistema operativo, la distribución y el método de instalación varía.

### 3.1.1. Instalación de $\LaTeX$ y $\TeX$

#### $\LaTeX$ en Windows

La versión más popular de  $\LaTeX$  para Windows se llama  $\text{MiK}\TeX$  y la puedes bajar desde <http://www.miktex.org>. Desde ahí bajas un Setup Wizard que, una vez instalado, se conecta a internet para bajar e instalar el resto del programa. Realmente se instala una versión reducida (guiada por asistente al típico estilo Windows), pero luego se pueden descargar aquellos módulos  $\text{LaTeX}$  (denominados paquetes) que se vayan necesitando mediante  $\text{MiK}\TeX$  Options, que viene incluido en la distribución. La distribución de  $\text{MiK}\TeX$  se puede ir actualizando mediante  $\text{MiK}\TeX$  Update Wizard.

Es muy recomendable, si quieres generar y ver tu tesis en el formato PDF, que tengas instalado Adobe Acrobat Reader en tu ordenador. Lo más normal es que ya lo tengas instalado pero, si no lo tienes, puedes bajarlo desde <http://www.adobe.com/products/acrobat>.

Necesitas también los programas AFPL Ghostscript y GSview para poder manipular archivos PostScript. Ambos programas los puedes conseguir en la página de Internet <http://www.cs.wisc.edu/~ghost/>.

Por último, también es muy recomendable que bajes el  $\text{T}\TeX\text{nicCenter}$ . Es un editor de texto especializado para  $\LaTeX$  con botones y ventanas, muy intuitivo y fácil de usar. Este programa, altamente recomendable, lo puedes bajar en la dirección <http://www.toolscenter.org/products/texniccenter/>.

Es importante que el último programa que instales sea  $\text{T}\TeX\text{nicCenter}$ . Ya que, al iniciarlo la primera vez, buscará donde tienes instalados  $\text{MiK}\TeX$  y el resto de las aplicaciones para configurar todas las opciones necesarias de manera automática. Una vez instalado  $\text{T}\TeX\text{nicCenter}$ , sólo hay que realizar unos sencillos pasos de configuración, consistentes en confirmar la ruta de acceso a  $\text{MiK}\TeX$  e indicar que se va a usar PDF (preferible, aunque también puede ser DVI o PS) como formato de salida, esto hará que  $\text{T}\TeX\text{nicCenter}$  ejecute automáticamente el comando  $\LaTeX$  apropiado.

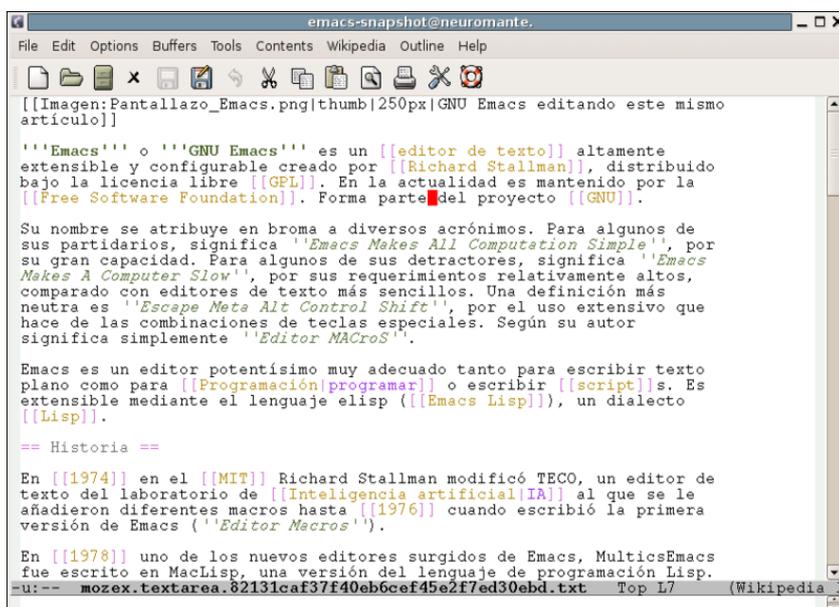


Figura 3.1: Emacs

## $\LaTeX$ en Linux

Hay que instalar el paquete TeTeX. Éste incluye todo lo necesario, excepto el editor para poder escribir los documentos LaTeX. Sin embargo el paquete TeTeX no ha sido mantenido en mucho tiempo. Esto ha llevado a buscar una solución, ésta se llama TexLive, que incorpora soluciones a bugs y mejoras respecto a su antecesor. Como editor puede usarse:

- **Emacs** (paquete emacs), que dispone de un modo de edición especial para  $\LaTeX$ , realizando los comandos. Puede ser conveniente evaluar una extensión para emacs denominada AUCTeX, que indenta automáticamente, con lo cual se obtiene una mejora ostensible de la legibilidad del código, entre otras cosas.
- **Kile**, en el cual disponemos de autocompletado de comandos  $\LaTeX$ , coloreado de sintaxis, Kile automáticamente marca los comandos  $\LaTeX$  y resalta los parentesis, y puede trabajar con múltiples ficheros a la vez. Además también proporciona plantillas y patrones para facilitar la creación de documentos.

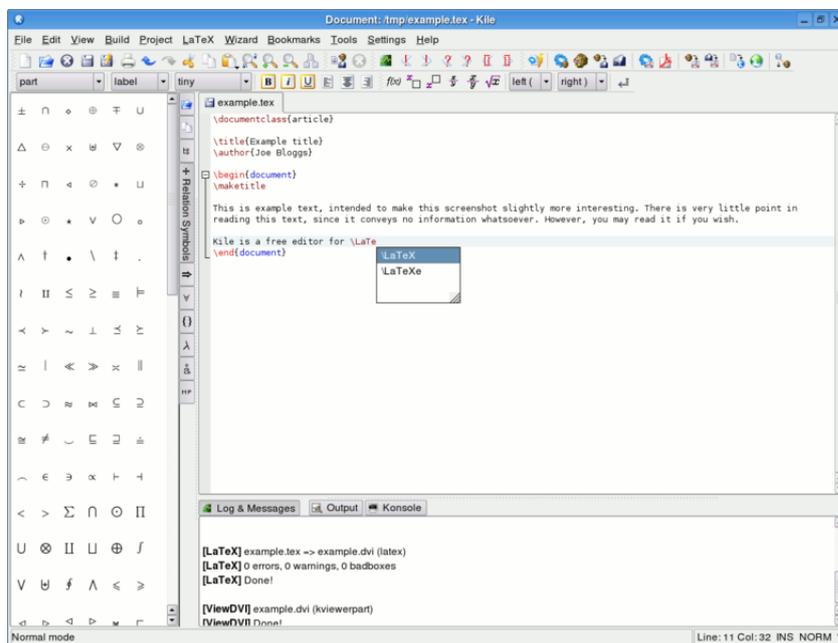


Figura 3.2: Kile

## 3.2– Lo que necesita saber sobre $\text{\LaTeX}$

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

El fichero de entrada para  $\text{\LaTeX}$  es un fichero de texto en formato ASCII. Se puede crear con cualquier editor de textos. Contiene tanto el texto que se debe imprimir como las instrucciones, con las cuales  $\text{\LaTeX}$  interpreta cómo debe disponer el texto.

#### Signos de espacio

Los caracteres invisibles, como el espacio en blanco, el tabulador y el final de línea, son tratados por  $\text{\LaTeX}$  como signos de espacio propiamente dichos. Varios espacios seguidos se tratan como un espacio en blanco. Generalmente, un espacio en blanco al comienzo de una línea se ignora, y varios renglones en blanco se tratan como un renglón en blanco.

Un renglón en blanco entre dos líneas de texto definen el final de un párrafo. Varias líneas en blanco se tratan como una sola línea en blanco. El texto que mostramos a continuación es un ejemplo. A la derecha se encuentra el texto del fichero de entrada y a la izquierda la salida formateada.

No importa si introduce varios espacios tras una palabra.

Con una línea vacía se empieza un nuevo párrafo.

No importa si introduce varios espacios tras una palabra.

Con una línea vacía se empieza un nuevo párrafo.

### Caracteres especiales

Los símbolos siguientes son caracteres reservados que tienen un significado especial para  $\LaTeX$  o que no están disponibles en todos los tipos. Si los introduce en su fichero directamente es muy probable que no se impriman o que fuercen a  $\LaTeX$  a hacer cosas que Vd. no desea.

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

Como puede ver, algunos de estos caracteres se pueden incluir en sus documentos anteponiendo el carácter (backslash) `\`:

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

Los restantes símbolos y otros muchos caracteres especiales se pueden imprimir en fórmulas matemáticas o como acentos con órdenes específicas.

### Las órdenes de $\LaTeX$

En las órdenes  $\LaTeX$  se distinguen las letras mayúsculas y las minúsculas. Toman uno de los dos formatos siguientes:

- Comienzan con un backslash y tienen un nombre compuesto sólo por letras. Los nombres de las órdenes acaban con uno o más espacios en blanco, un carácter especial o una cifra.
- Se compone de un backslash y un carácter especial.

$\LaTeX$  ignora los espacios en blanco que van tras las órdenes. Si se desea introducir un espacio en blanco tras una instrucción, se debe poner o bien `{ }` y un espacio, o bien una instrucción de espaciado después de la orden. Con `{ }` se fuerza a  $\LaTeX$  a dejar de ignorar el resto de espacios que se encuentren después de la instrucción.

He leído que Knuth distingue a la gente que trabaja con  $\TeX$  en  $\TeX$ nicos y  $\TeX$ pertos.

Hoy es 4 de septiembre de 2008.

He le'ido que Knuth distingue a la gente que trabaja con `\TeX{}` en `\TeX{}`nicos y `\TeX{}`pertos.`\`  
Hoy es `\today`.

Algunas instrucciones necesitan un parámetro que se debe poner entre llaves `{}` tras la instrucción. Otras órdenes pueden llevar parámetros opcionales que se añaden a la instrucción entre corchetes `[]` o no.

### Comentarios

Cuando  $\text{\LaTeX}$  encuentra un carácter `%` mientras procesa un fichero de entrada, ignora el resto de la línea. Esto suele ser útil para introducir notas en el fichero de entrada que no se mostrarán en la versión impresa.

Esto es un ejemplo.	<pre>Esto es un % tonto % o mejor instructivo ejemplo.</pre>
---------------------	--

Esto a veces puede resultar útil cuando nos encontramos con líneas demasiado largas en el fichero fuente. Si no quisiésemos introducir un espacio entre dos palabras, y preferimos tener dos renglones, entonces el signo `%` debe ir justo al final del renglón pero pegado al último carácter. De este modo comentamos el carácter de salto de línea, que se hubiese tratado como un espacio en blanco.

Este es otro ejemplo.	<pre>Este es otro ejem% y % ahora el resto plo.</pre>
-----------------------	---

### 3.2.2. Estructura de un fichero de entrada

Cuando  $\text{\LaTeX 2}_\epsilon$  procesa un fichero de entrada, espera de él que siga una determinada estructura. Todo fichero de entrada debe comenzar con la orden

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

Esto indica qué tipo de documento es el que se pretende crear. Tras esto, se pueden incluir órdenes que influirán sobre el estilo del documento entero, o cargar paquetes que añadirán nuevas propiedades al sistema de  $\text{\LaTeX}$ . Para cargar uno de estos paquetes se usará la instrucción

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

Cuando todo el trabajo de configuración esté realizado entonces comienza el cuerpo del texto con la instrucción

```
\begin{document}
```

El área comprendida entre `\documentclass{...}` y `\begin{document}` recibe el nombre de preámbulo, una vez que éste ha finalizado se introducirá el texto mezclado con algunas instrucciones útiles de  $\text{\LaTeX}$ . Por último para finalizar el documento debe ponerse la orden

```
\end{document}
```

De esta forma  $\text{\LaTeX}$  ignorará cualquier cosa que se ponga tras esta instrucción. Las figuras 3.3 y 3.4 muestran el contenido mínimo de un fichero de  $\text{\LaTeX} 2_{\epsilon}$ , así como un fichero de entrada algo más complicado.

---

```
\documentclass{article}
\begin{document}
Lo bueno si es breve, dos veces bueno.
\end{document}
\end{verbatim}
```

---

Figura 3.3: Fichero mínimo de  $\text{\LaTeX}$

---

```
\documentclass[a4paper,11pt]{article}
\usepackage{latexsym}
\usepackage[activeacute,spanish]{babel}
\author{H. ~Partl}
\title{Minimizando}
\frenchspacing
\begin{document}
\maketitle
\tableofcontents
\section{Inicio}
Aqu'i comienza un art'iculo estupendo.
\section{Fin}
Aqu'i acaba mi art'iculo.
\end{document}
```

---

Figura 3.4: Ejemplo de artículo científico en español

### 3.2.3. El formato del documento

#### Clases de documentos

Cuando procesa un fichero de entrada, lo primero que necesita saber  $\text{\LaTeX}$  es el tipo de documento que el autor quiere crear. Esto se indica con la instrucción

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

En este caso, la clase indica el tipo de documento que se creará. En el cuadro 3.1 se muestran las clases de documento que explicaremos más adelante. La distribución de  $\text{\LaTeX} 2_{\epsilon}$  proporciona más clases para otros documentos, como cartas y transparencias. El parámetro de opciones personaliza el comportamiento de la clase de documento elegida. Las opciones

se deben separar con comas. En el cuadro 3.2 se indican las opciones más comunes de las clases de documento estándares.

Por ejemplo: un fichero de entrada para un documento de  $\text{\LaTeX}$  podría comenzar con

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

Esto le indica a  $\text{\LaTeX}$  que componga el documento como un artículo utilizando tipos del cuerpo 11, y que produzca un formato para impresión a doble cara en papel DIN A4.

**article** para artículos de revistas especializadas, ponencias, trabajos de prácticas de formación, trabajos de seminarios, informes pequeños, solicitudes, dictámenes, descripciones de programas, invitaciones y muchos otros.

**report** para informes mayores que constan de más de un capítulo, proyectos fin de carrera, tesis doctorales, libros pequeños, disertaciones, guiones y similares.

**book** para libros de verdad.

**slide** para transparencias. Esta clase emplea tipos grandes sans serif.

Cuadro 3.1: Clases de documentos

## Paquetes

En algunas situaciones el  $\text{\LaTeX}$  básico no es suficiente, por ejemplo si queremos incluir gráficos, texto en color o el código fuente de un fichero, necesita mejorar las capacidades de  $\text{\LaTeX}$ . Tales mejoras son conocidas como paquetes. Los paquetes se activan con la orden

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

donde paquete es el nombre del paquete que queremos usar y opciones es una lista palabras clave que activan funciones especiales del paquete. Algunos paquetes vienen con la distribución básica de  $\text{\LaTeX}$  (véase la Cuadro 3.3). Otros se porporcionan por separado. En la Guía Local [gui] puede encontrar más información sobre los paquetes disponibles en su instalación local. La fuente principal de información sobre  $\text{\LaTeX}$  es The  $\text{\LaTeX}$  Companion [MG94]. Contiene descripciones de cientos de paquetes, así como información sobre cómo escribir sus propias extensiones a  $\text{\LaTeX 2}_\epsilon$ .

---

**10pt, 11pt, 12pt** Establecen el tamaño (cuerpo) para los tipos. Si no se especifica ninguna opción, se toma **10pt**.

**a4paper, letterpaper, ...** Define el tamaño del papel. Si no se indica nada, se toma **letterpaper**. Aparte de éste se puede elegir **a5paper**, **b5paper**, **executivepaper** y **legalpaper**.

**fleqn** Dispone las ecuaciones hacia la izquierda en vez de centradas.

**leqno** Coloca el número de las ecuaciones a la izquierda en vez de a la derecha.

**titlepage, notitlepage** Indica si se debe comenzar una página nueva tras el título del documento o no. Si no se indica otra cosa, la clase **article** no comienza una página nueva, mientras que **report** y **book** sí.

**twocolumn** Le dice a L<sup>A</sup>T<sub>E</sub>X que componga el documento en dos columnas.

**twoside, oneside** Especifica si se debe generar el documento a una o a dos caras. En caso de no indicarse otra cosa, las clases **article** y **report** son a una cara y la clase **book** es a dos.

**openright, openany** Hace que los capítulos comiencen o bien sólo en páginas a la derecha, o bien en la próxima que esté disponible. Esto no funciona con la clase **article**, ya que en esta clase no existen capítulos. De modo predeterminado, la clase **report** comienza los capítulos en la próxima página disponible y la clase **book** las comienza en las páginas a la derecha.

- 
- doc** Permite la documentación de paquetes y otros ficheros de  $\text{\LaTeX}$ .  
Se describe en `doc.dtx`.
- exscale** Proporciona versiones escaladas de los tipos adicionales para matemáticas.  
Descrito en `ltxscale.dtx`.
- fontenc** Especifica qué codificación de tipo debe usar  $\text{\LaTeX}$ .  
Descrito en `ltoutenc.dtx`.
- ifthen** Proporciona instrucciones de la forma ‘si . . . entonces. . . si no. . .’,  
Descrito en `ifthen.dtx`.
- latexsym** Para que  $\text{\LaTeX}$  acceda al tipo de símbolos, se debe usar el paquete `latexsym`.  
Descrito en `latexsym.dtx`.
- makeidx** Proporciona instrucciones para producir índices de materias.
- syntonly** Procesa un documento sin componerlo, lo cual es útil para la verificación rápida de errores.  
Se describe en `syntonly.dtx`.
- inputenc** Permite la especificación de una codificación de entrada como ASCII (con la opción `ascii`), ISO Latin-1 (con la opción `latin1`), ISO Latin-2 (con la opción `latin2`), páginas de código de 437/850 IBM (con las opciones `cp437` y `cp580`, respectivamente), Apple Macintosh (con la opción `applemac`), Next (con la opción `next`), ANSI-Windows (con la opción `ansinew`) o una definida por el usuario.  
Descrito en `inputenc.dtx`.
- 

Cuadro 3.3: Algunos paquetes distribuidos con  $\text{\LaTeX}$

### Estilo de página

Con  $\text{\LaTeX}$  existen tres combinaciones predefinidas de cabeceras y pies de página, a las que se llaman estilos de página. El parámetro estilo de la instrucción

```
\pagestyle{estilo}
```

define cuál usarse. La Cuadro 3.4 muestra los estilos de página predefinidos.

---

**plain** imprime los números de página en el centro del pie de las páginas.

Este es el estilo de página que se toma si no se indica ningún otro.

**headings** en la cabecera de cada página imprime el capítulo que se está procesando y el número de página, mientras que el pie está vacío.

(Este estilo es similar al empleado en este documento).

**empty** deja tanto la cabecera como el pie de las páginas vacíos.

---

Cuadro 3.4: Estilos de página predefinidos en  $\text{\LaTeX}$

Es posible cambiar el estilo de página de la página actual con la instrucción

```
\thispagestyle{estilo}
```

En *The  $\text{\LaTeX}$  Companion* [3] hay una descripción de cómo crear sus propias cabeceras y pies de página.

#### 3.2.4. Proyectos grandes

Cuando trabaje con documentos grandes, podría, si lo desea, dividir el fichero de entrada en varias partes.  $\text{\LaTeX}$  tiene varias instrucciones que le ayudan a realizar esto.

```
\include{fichero}
```

Se puede utilizar en el cuerpo del documento para introducir el contenido de otro fichero. Observe que  $\text{\LaTeX}$  comenzará una página nueva antes de procesar el texto del fichero.

La siguiente instrucción solo puede ser utilizada en el preámbulo. Permite indicarle a  $\text{\LaTeX}$  que sólo tome la entrada de algunos ficheros de los indicados con `\include`.

```
\includeonly{fichero, fichero,...}
```

Una vez que esta instrucción se ejecute en el preámbulo del documento, sólo se procesarán las instrucciones `\include` con los ficheros indicados en el argumento de la orden `\includeonly`. Observe que no hay espacios entre los nombres de fichero y las comas.

Existe otra opción que puede ser usada a lo largo del documento para introducir el texto situado en el fichero correspondiente. En este caso no se produce ningún salto de página ni de línea a menos que se indique adecuadamente.

```
\input{fichero}
```

Resulta muy útil para dividir un trabajo en diferentes archivos y agruparlos en un documento donde tendremos un preámbulo común a todos.

### 3.3– ¿Qué documentación hay sobre T<sub>E</sub>X/L<sub>A</sub>T<sub>E</sub>X?

#### 3.3.1. Tutoriales

##### Tutoriales en castellano

- **Una descripción de L<sub>A</sub>T<sub>E</sub>X** Tomás Bautista. Este documento se encuentra en CTAN (en CTAN:documentation/short/spanish)
- **Bases de datos bibliográficos, L<sub>A</sub>T<sub>E</sub>X y el idioma español** Luis Seidel: <ftp://tex.unirioja.es/pub/tex/doc/bibliogr.pdf>
- **Recetario para L<sub>A</sub>T<sub>E</sub>X** por Aristarco. Disponible en: <http://recetariolatex.cjb.net>

##### Tutoriales en otros idiomas

- **A Gentle Introduction to T<sub>E</sub>X** de Michael Dobb, disponible en :CTAN:documentation/gentle
- **Simplified Introduction to L<sub>A</sub>T<sub>E</sub>X** de Harvey J. Greenberg, disponible en :CTAN:documentation/simplified-latex/latex.ps
- **The not so Short Introduction to L<sub>A</sub>T<sub>E</sub>X** de Tobias Oetiker, disponible en :CTAN:documentation/lshort/

#### 3.3.2. Libros

##### Libros en castellano

- **El libro de L<sub>A</sub>T<sub>E</sub>X** [BC03]. Bernardo Cascales, Pascual Lucas, José Manuel Mira, Antonio Pallarés y Salvador Sánchez-Pedreño. Prentice Hall, Madrid, 2003. Se divide en dos partes: en la primera, que consta de 18 lecciones, el lector encontrará todo lo que necesita para componer documentos básicos con L<sub>A</sub>T<sub>E</sub>X, que incluso pueden llegar a alcanzar una complejidad tipográfica notable. En la segunda parte, que tiene intención de servir como manual de referencia y uso avanzados, profundiza en todos los aspectos tratados en las lecciones y añade otros nuevos.

- **Iniciación a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>** [Bot97]. Javier Sanguino Botella, Addison-Wesley (1997). Uno de los mejores manuales para empezar a aprender.
- **Composición de textos científicos con L<sup>A</sup>T<sub>E</sub>X** [Val97]. G. Valiente. Edicions UPC, Barcelona, 1997. Pretende dar a conocer el sistema L<sup>A</sup>T<sub>E</sub>X en el contexto de la composición de textos científicos. Así, puede servir de introducción a aquellos estudiantes que se inicien en la escritura científica y, a la vez, puede resultar un texto de consulta permanente para profesores e investigadores. Instructivo y fácil de leer, puede adquirirse en versión papel o en formato electrónico, a través del web de la editorial Edicions UPC [www.edicionsupc.es](http://www.edicionsupc.es).
- **L<sup>A</sup>T<sub>E</sub>X, una imprenta en sus manos** [Cas00]. Bernardo Cascales Salinas. Aula Documental de Investigación, 2000. Está escrito a modo enciclopédico y toda la información sobre un tema está tratada en un único lugar. Extensa obra que describe tanto los comandos básicos como las funcionalidades más avanzadas de L<sup>A</sup>T<sub>E</sub>X. Se incluye una descripción de la programación de macros y generación de documentos para su publicación en Internet. Se muestran interesantes ejemplos de uso en campos alejados de las matemáticas, como puede ser la composición tipográfica en otros alfabetos, símbolos químicos e incluso partituras musicales.

#### Libros en otros idiomas

- **A Guide to L<sup>A</sup>T<sub>E</sub>X** [?]. H. Kopka y P.W. Daly, Addison-Wesley Professional (2004). Probablemente, el mejor manual existente sobre L<sup>A</sup>T<sub>E</sub>X. Contiene una guía completa de órdenes, abundantes ejemplos e información adicional. (Incluye las dos versiones en uso de L<sup>A</sup>T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> y la más antigua, L<sup>A</sup>T<sub>E</sub>X2.09).
- **The L<sup>A</sup>T<sub>E</sub>X Companion**. [MG94]. Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, Addison Wesley Professional (2004). Este manual sirve de ampliación a [?]. Es una recopilación de información sobre los llamados paquetes (packages), conjuntos de macros que distintos autores han puesto a disposición pública.
- **L<sup>A</sup>T<sub>E</sub>X-A Document preparation system**[Lam94]. L. Lamport (dos ediciones) Addison-Wesley (1985 y 1994). Escrito por Leslie Lamport (autor de L<sup>A</sup>T<sub>E</sub>X) esto lo dice todo, contiene todo lo necesario para iniciarse en el mundo L<sup>A</sup>T<sub>E</sub>X. Puede resultar un tanto insuficiente para usuarios avanzados.

### 3.3.3. Varios

**CervanTeX** Grupo de usuarios hispanohablantes de TeX. Esta asociación busca intercambiar experiencias sobre TeX y L<sup>A</sup>TeX, así como sobre sus aplicaciones, y promoverlo de forma adecuada en el ámbito hispanohablante (tanto España como América). <http://www.cervantex.es>

**CTAN: The Comprehensive TeX Archive Network** Esta página es un repositorio de software y documentación relacionada con TeX. <http://www.ctan.org/>

**The PracTeX Journal** Una revista de TeX online y gratuita. Disponible en <http://dw.tug.org/pracjourn/>

**LaTeX-project** Aquí encontrarás material de referencia para aprender a utilizar y sacarle más provecho a tu sistema de L<sup>A</sup>TeX. <http://www.LaTeX-project.org>

## CAPÍTULO 4

---

### Análisis de requisitos, diseño e implementación

---

A lo largo de este Capítulo 4 describiremos por un lado el diseño e implementación de la clase `pclass.cls`. Resulta sencillo adivinar por su extensión `.cls` que está implementada en lenguaje  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ , esto se debe a de que dicho lenguaje reserva esta extensión para sus clases. Por otro lado también tendrá cabida en este capítulo un apartado dedicado al Análisis de Requisitos. Haremos uso de dicho apartado para especificar los requisitos mínimos que a nuestro juicio debe cumplir una clase, teniendo dicha clase como objetivo el formateo de memorias relativas a proyectos de fin de carrera.

Sin embargo además de los requisitos mínimos citados en el párrafo anterior, también se hará referencia a funcionalidades adicionales que se han añadido a través del formato definido en `pclass`. Todas estas funcionalidades que se han añadido tendrán un mismo objetivo, simplificar al máximo al usuario de la clase `pclass` todas las tareas que conlleva la redacción de una memoria de proyecto de final de carrera.

#### 4.1– Diseño e implementación

En lo que respecta al Diseño de la clase que da origen a este proyecto, resultará necesaria inicialmente la participación del tutor del proyecto, José Ramón Portillo Fernández. De este modo el tutor nos indicará una serie de referencias para poder iniciar el diseño e implementación de nuestra propia clase  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ . Esta participación se fundamenta en dos motivos principalmente:

1. En el hecho de que la temática de este proyecto sea una de las propuestas ofertadas por el departamento de Matemática Aplicada

para el curso 2007/2008, llevada a cabo por José Ramón Portillo Fernández.

2. La carencia de conocimientos previos de los autores de este proyecto acerca del lenguaje de programación  $\text{\LaTeX}$ . Este hecho hará que la creación de la clase `pclass` constituya nuestro primer contacto con el citado lenguaje.

Como el título de este proyecto indica el lenguaje utilizado para implementar la clase `pclass` es  $\text{\LaTeX}$ . Esta decisión se basa en las innumerables ventajas que puede ofrecer a sus usuarios dicho lenguaje. Para conocer más acerca de esta serie de ventajas puedes hechar un vistazo al Capítulo 2 de esta memoria o recurrir a fuentes bibliográficas como [Gal92].

Es importante resaltar que la implementación de `pclass` la hemos realizado simultáneamente en dos sistemas operativos. Estos sistemas operativos son concretamente Windows y Linux, habiendo usado en cada uno de ellos el entorno de desarrollo descrito más adelante. La razón de este paralelismo en la implementación no es mas que conocer de primera mano los distintos inconvenientes que derivan de cada una de las plataformas usadas. De esta forma nos beneficiamos del conocimiento de ambas plataformas, y podemos ofrecer al usuario de la clase `pclass` una solución a los problemas que surgen de forma común.

En lo relativo al entorno de desarrollo empleado diferenciaremos evidentemente entre el usado para Windows y el respectivo para Linux.

Empezaremos describiendo el entorno de desarrollo empleado bajo Windows:

**MiKTeX** Es la versión más popular de  $\text{\LaTeX}$  para Windows y la puedes bajar desde <http://www.miktex.org>. Desde ahí bajas un *Setup Wizard* que, una vez instalado, se conecta a internet para bajar e instalar el resto del programa.

**TeXnicCenter** Es un editor de texto especializado para  $\text{\LaTeX}$  con botones y ventanas, muy intuitivo y fácil de usar. Este programa, altamente recomendable, lo puedes bajar en la dirección <http://www.toolscenter.org/products/teXniccenter/>.

**Adobe Acrobat Reader** Visor de documentos recomendable si quieres generar y ver tu memoria en el formato PDF (*Portable Document Format*). Puede encontrarse en la dirección: <http://www.adobe.com/products/acrobat>.

**AFPL Ghostscript y GSview** Ambos son visores de documentos que te permitirán manipular archivos *PostScript*. Ambos programas los puedes conseguir en la página de Internet <http://www.cs.wisc.edu/~ghost/>.

En el caso de Linux las descripciones del entorno de desarrollo usado son:

**TexLive** Se trata de una actualización del paquete TeTeX. Incluye todo lo necesario, excepto el editor, para poder escribir los documentos L<sup>A</sup>T<sub>E</sub>X. Además incorpora soluciones a bugs y mejoras respecto a su antecesor. Puedes encontrarlo en <http://www.tug.org/texlive/acquire.html>.

**Kile** Editor de texto para L<sup>A</sup>T<sub>E</sub>X en el cual disponemos de autocompletado de comandos, coloreado de sintaxis, marca automáticamente los comandos L<sup>A</sup>T<sub>E</sub>X y resalta los paréntesis, y puede trabajar con múltiples ficheros a la vez. Además también proporciona plantillas y patrones para facilitar la creación de documentos.

**Adobe Acrobat Reader** Al igual que para windows es un visor de documentos para generar y ver tu memoria en el formato PDF.

## 4.2— Análisis de requisitos

En este apartado de Análisis de Requisitos enunciaremos una serie de requisitos que cumple la clase `pclass`. Todos estos requisitos perseguirán una única meta, la consecución de una memoria de proyecto de fin de carrera que reúna todas las especificaciones relativas al formato característico de este tipo de documentos.

Además de cumplir los requisitos relativos al formato del documento, la clase `pclass` también posee diversas funcionalidades encaminadas a facilitar al usuario de la clase el cumplimiento de estos requisitos de formato. Por tanto esta serie de funcionalidades, como no puede ser de otra forma, también serán contempladas a lo largo del apartado que nos ocupa.

### Portada

Un requisito indispensable a la hora de elaborar la documentación de un proyecto de fin de carrera, es generar una portada adecuada. Cuando hablamos de una portada adecuada nos referimos a que ésta cumpla las directrices relativas al formato indicadas por la Universidad de Sevilla.

En L<sup>A</sup>T<sub>E</sub>X para la generación de este tipo de portadas hay que recurrir inevitablemente a una importante sucesión de comandos. Por este motivo en la clase `pclass` se ha definido una macro llamada `\hacerportada`, que contendrá la sucesión de comandos necesaria. De esta forma se facilita la tarea al usuario de la clase, ya que con un solo comando obtendrá una portada perfectamente formateada. El formato para la portada definido en `pclass` puedes visualizarlo en la Figura 4.1.



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**(NOMBRE DE LA TITULACIÓN)**

**(TÍTULO DEL PROYECTO)**

**Realizado por  
(NOMBRE DEL ALUMNO)**

**Dirigido por  
(NOMBRE DEL TUTOR DEL PROYECTO)**

**Departamento  
(NOMBRE DEL DEPARTAMENTO)**

**Sevilla, (mes y año)**

Figura 4.1: Formato definido en pclass para la portada

Sin embargo para generar la portada el usuario deberá indicar una serie de datos. Éstos serán usados con el fin de rellenar la portada con la información adecuada en cada caso. Para especificar esta información el usuario tendrá que completar al comienzo del archivo `proyect.tex` los comandos que podemos observar en el código siguiente:

```

1 \documentclass{pclass}
2 \begin{document}
3 \titulopro{ }% Título del proyecto
4 \tutor{ }% Nombre del tutor del proyecto
5 \departamento{ }% Departamento al que pertenece tu proyecto
6 \autores{ }% Nombre de los autores
7 \dia{mm/aaaa}% fecha de entrega de la memoria
8 \titulacion{ }% Titulacion cursada por los autores

```

Código 4.1: Datos para completar la portada

Si tienes alguna duda sobre como completar estos comandos puedes echar un vistazo al Capítulo 11.

Hemos mencionado ya que la macro `\hacerportada` esconde tras de sí una sucesión de comandos para generar una portada con el formato adecuado. Esta sucesión de comandos puedes visualizarla en el código mostrado mas adelante. Si quieres conocer más acerca de la finalidad de cada uno de estos comandos puedes hacerlo en el Capítulo 9.

```

1 \newcommand{\hacerportada}{
2 \begin{titlepage}
3 \centering
4 \includegraphics[scale=2.5]{img/us}
5 \begin{center}\bf\sffamily
6 {\normalsize ESCUELA TECNICA SUPERIOR DE INGENIERIA INFORMATICA
7 }\\[1cm]
8 {\large \@titulacion{}}\\[2.5cm]
9 {\large \@titulopro{}}\\[2cm]
10 {\large Realizado por}\\
11 {\large \@autorUno{}}\\
12 {\large \@autorDos{}}\\
13 [1cm]
14 {\large Dirigido por}\\
15 {\large \@tutor}\\[1cm]
16 {\large Departamento}\\
17 {\large \@departamento}\\[3cm]
18 \end{center}
19 \begin{flushright}
20 {\bf\sffamily\large Sevilla, ({\large \@diad})}
21 \end{flushright}
22 \end{titlepage}
23 }

```

Código 4.2: Macro para hacer la portada

## Resumen

Toda memoria que se precie debe contar al inicio de la misma con un resumen del proyecto. Con el fin de recordárselo al usuario de la clase `pclass`, dentro del archivo `project.tex` en la sección perteneciente a `\frontmatter` puedes encontrar la línea `\input{resumen.tex}`. Esta línea se encargará de insertar en la memoria un `.tex` que contiene el resumen del proyecto. Puedes localizar la inserción del resumen en la captura de Código: Contenido de frontmatter.

```
1 \frontmatter
2 % Introduce como resumen el contenido de resumen.tex
3 \input{resumen.tex}
4 % Introduce como agradecimientos
5 % el contenido de agradecimientos.tex
6 \input{agradecimientos.tex}
7 %Indice de contenidos
8 \tableofcontents
9 % Indice de tablas
10 \listoftables
11 % Indice de figuras
12 \listoffigures
13 % Indice de capturas de codigo
14 \lstlistoflistings
```

Código 4.3: Contenido de frontmatter

## Agradecimientos

Al igual que ocurre con el Resumen, toda memoria debe contener unas páginas para agradecimientos. Con el fin de recordárselo al usuario de la clase `pclass`, dentro del archivo `project.tex` en la sección perteneciente a `\frontmatter` puedes encontrar la línea `\input{agradecimientos.tex}`. Este comando insertará al comienzo de la memoria una sección dedicada a los agradecimientos. Puedes localizar la inserción de los agradecimientos en la captura de Código: Contenido de frontmatter.

## Índice General

Un requisito indispensable para una memoria de proyecto es la existencia de un índice general de contenidos. Para que dicho índice aparezca en la memoria, se hace uso del comando `\tableofcontents`. Puedes localizar este comando en la sección perteneciente a `\frontmatter` dentro de `project.tex`. Para verlo más claro puedes ver la captura Código: Contenido de frontmatter.

## Índice de cuadros y Figuras

Resulta también indispensable la existencia de un índice de cuadros y otro de figuras. Usamos los comandos `\listoftables` y `\listoffigures`, dentro del frontmatter de `proyect.tex`, para generar ambos índices. Puedes verlo con mayor claridad en la captura Código: Contenido de frontmatter.

Para facilitar las tareas de inserción de cuadros y figuras, las cuales quedan reflejadas en los índices mencionados, hemos definido en nuestra clase una macro para cada una de estas tareas. Concretamente se trata de las macros `\cuadro` y `\figura`, puedes ver el código con el que son definidas en la captura siguiente.

```

1
2 % 1 --> Porcentaje del ancho de pagina que ocupara la figura (de 0
   a 1)
3 % 2 --> Fichero de la imagen
4 % 3 --> Texto a pie de imagen
5 % 4 --> Etiqueta (label) para referencias
6 % 5 --> Opciones que queremos pasarle al \includegraphics
7 \newcommand{\figura}[5]{%
8   \begin{figure}%
9     \begin{center}%
10      \includegraphics[width=#1\textwidth,#5]{#2}%
11      \caption{#3}%
12      \label{#4}%
13    \end{center}%
14  \end{figure}%
15 }
16 %1 ---> especificar numero de columnas y alineacion
17 % ejm: |r|c|c| r=right, c=center,l=left
18 %2 ---> especificar el caption o titulo de la figura
19 %3 ---> label para hacer referencia a la tabla insertada
20 %4 ---> contenido de la tabla separando columnas con & y filas con
    \\
21 \newcommand{\tabla}[4]{
22   \begin{table}[htb]
23     \centering
24     \begin{tabular}{#1}
25       \hline
26       #4
27       \hline
28     \end{tabular}
29     \caption{#2}
30     \label{#3}
31   \end{table}
32 }
```

Código 4.4: Macros cuadro y figura

## Índice de código

No es una condición imprescindible la aparición en una memoria de un índice de capturas de código. Sin embargo en los proyectos de titulaciones pertenecientes a la Escuela Técnica Superior de Ingeniería Informática, aparecen una cantidad importante de capturas de código. Por este motivo hemos considerado que puede ser de gran utilidad definir un índice de capturas de código.

Para que quede reflejado este índice en la memoria se incluye el comando `\lstlistoflistings` en el frontmatter del archivo `proyect.tex`, puedes localizar esta instrucción en la captura Código: Contenido de frontmatter. Además de esto para materializar esta definición se han incluido las sentencias mostradas a continuación dentro de nuestra clase `pclass.cls`.

```

1  % Para generar indice de codigo fuente
2
3  \renewcommand{\lstlistlistingname}{Indice de Codigo}
4
5  \renewcommand{\lstlistingname}{Codigo}
6
7  %%%%%%%%%%%%%%% Para codigo fuente %%%%%%%%%%
8
9  \newcommand{\codigofuente}[3]{%
10   \lstinputlisting[language=#1,caption={#2}]{#3}%
11 }
12 %%%%%%%%%%%%%%%
13 %%%%%%%%%%%%%%%

```

Código 4.5: Comandos para generar el índice de código

Por otra parte también hemos querido facilitar al máximo la inserción de capturas de código fuente. Para ello hemos recurrido al paquete `listings`, definiendo basándonos en él la macro `\codigofuente`.

Con esta macro conseguimos que el usuario inserte fácilmente capturas consistentes en el contenido de archivos `LATEX`, de igual forma podremos hacerlo con archivos que contengan muchos otros lenguajes de programación. Para insertar una captura bastará con hacer uso de la macro citada, indicando en sus argumentos el nombre del archivo deseado, el lenguaje de programación usado en el mismo y el título con el que aparecerá la captura.

Si quieres conocer algo más acerca de este tema puedes encontrar más información en el Capítulo 9. En la captura siguiente puedes encontrar todas las definiciones realizadas en `pclass` relativas al paquete `listings`.

```
1 % Definiendo colores para los listados de código
2 \usepackage{listings}
3
4 \definecolor{violet}{rgb}{0.5,0,0.5}
5 \definecolor{shadow}{rgb}{0.5,0.4,0.5}
6 \definecolor{hellgelb}{rgb}{1,1,0.8}
7 \definecolor{colKeys}{rgb}{0.6,0.15,0}
8 \definecolor{colIdentifier}{rgb}{0.7,0.1,0}
9 \definecolor{colComments}{cmyk}{0,0.3,0.99,0.25}
10 \definecolor{colString}{rgb}{0,0.5,0}
11
12 \lstset{
13     framexleftmargin=5mm,
14     frame=shadowbox,
15     rulesepcolor=\color{shadow},
16     float=hbp,
17     basicstyle=\ttfamily\small,
18     identifierstyle=\color{colIdentifier},
19     keywordstyle=\color{colKeys},
20     stringstyle=\color{colString},
21     commentstyle=\color{colComments},
22     columns=flexible,
23     tabsize=4,
24     extendedchars=true,
25     showspace=false,
26     showstringspaces=false,
27     numbers=left,
28     numberstyle=\tiny,
29     breaklines=true,
30     backgroundcolor=\color{hellgelb},
31     breakautoindent=true,
32     captionpos=b
33 }
34 % Indices de código fuente
35 \renewcommand{\lstlistingname}{Índice de Código}
36 \renewcommand{\lstlistingname}{Código}
37
38 % 1 ---> Lenguaje segun la tabla de opciones.
39 % 2 ---> título de la captura de código
40 % 3 ---> Ruta del archivo
41 % 4 ---> Etiqueta para referenciar la captura
42 \newcommand{\codigofuente}[3]{%
43     \lstinputlisting[language=#1,caption={#2}]{#3}
44 }
```

Código 4.6: Definiciones relativas al paquete listings

## Apéndices

Resulta imprescindible que en toda memoria se reserve un espacio dedicado a posibles apéndices. Para ello en el archivo `project.tex` realizaremos, siempre tras el comando `\backmatter` y antes de `\end{document}`, tantos `\input` o `\includes` como necesitemos para la inclusión de los apéndices que sean necesarios. En nuestro caso hemos incluido en nuestro proyecto un apéndice dedicado a la licencia GNU, además de la bibliografía que también constituye un apéndice. Puedes verlo más claro en la captura siguiente.

```
1
2 %comienza el backmatter
3 \backmatter
4 % apéndice relativo a la licencia
5 \input{Capitulos/licencia}
6
7 % para generar la bibliografía
8 \bibliographystyle{pfc}
9 \bibliography{pfcbib}
10
11 \end{document}
```

Código 4.7: Apéndices

## Bibliografía

Otro de los apartados que resultan clave en una memoria de proyecto es el dedicado a referencias bibliográficas. No es necesario que el usuario se preocupe por la forma en que se escriben los datos de los libros en la sección de referencias bibliográficas,  $\text{\LaTeX}$  hace eso de manera automática.

Existen varios estilos bibliográficos predefinidos en  $\text{\LaTeX}$ , sin embargo paralelamente a `pclass` hemos creado nuestro propio estilo para la bibliografía. De esta forma a través de nuestro estilo que recibe el nombre de `pfcbibstyle`, podemos ajustar todos los aspectos referentes a la bibliografía según nuestras preferencias y necesidades. Para conocer más detalles acerca de la creación de `pfcbibstyle` y de las ventajas que nos proporciona puedes consultar el Capítulo 9.

Para incluir la sección de bibliografía en la memoria, serán necesarios una serie de comandos dentro del apartado `\backmatter` del archivo `project.tex`. Estos comandos son concretamente:

1. `\bibliographystyle{pfcbibstyle}`

Con este comando indicamos el estilo que queremos aplicar a nuestra bibliografía. En este caso, como ya hemos mencionado, se tratará del estilo `pfcbibstyle`.

## 2. `\bibliography{pfcbib}`

Con esta sentencia indicamos el nombre del archivo `.bib` que contiene los datos de todas nuestras referencias bibliográficas. En nuestro caso se tratará del archivo `pfcbib`. Este archivo podemos considerarlo como una base de datos de nuestros libros, a la que `LATEX` recurrirá para obtener todos los datos necesarios para generar la bibliografía.

En el fragmento de código mostrado seguidamente podrás visualizar claramente cómo se incluye la bibliografía en `proyect.tex`.

```
1
2 % comienza backmatter para apendices y bibliografia
3 \backmatter
4
5 % indicamos nuestro estilo bibliografico
6 \bibliographystyle{pfc}
7
8 % indicamos el archivo .bib utilizado
9 \bibliography{pfcbib}
10
11 \end{document}
```

Código 4.8: Comandos para incluir la bibliografía



## CAPÍTULO 5

---

### Análisis de antecedentes y aportación realizada

---

En primer lugar en este apartado debemos mencionar, que en la actualidad no existe ningún antecedente de clase  $\text{\LaTeX}$  para formatear proyectos de fin de carrera, para ninguna titulación perteneciente a la Universidad de Sevilla. Como consecuencia este proyecto supone una innovación en su ámbito dentro de dicha Universidad y resulta imposible la comparación de este proyecto con cualquier otro de carácter similar.

En cambio si que es posible hallar una extensa documentación de clases  $\text{\LaTeX}$  que llevan a cabo la misma labor, enfocadas a la presentación de memorias de Master's Thesis pertenecientes a diversas Universidades de los Estados Unidos de América. A pesar de no ser un referente para el proyecto que nos ocupa, si que podemos percibir que comparten algunos aspectos que nos pueden resultar de interés a la hora de implementar nuestra clase, aunque debemos salvar algunas barreras como puede ser adaptar la tipografía inglesa a la española. Para consultar dicho material podemos recurrir a la siguiente fuente <http://www.ctan.org>.

Puede resultar extremadamente difícil e infructuoso, encontrar una implementación de una clase  $\text{\LaTeX}$  que se encargue de formatear memorias de proyectos de fin de carrera en español. Sin embargo, tras un intenso periodo de búsqueda hemos conseguido localizar una clase implementada en este lenguaje que lleve a cabo la labor mencionada. Esta clase llamada `memoriaPFC.cls` puedes encontrarla en:

<http://websvn.eridem.net/listing.php?repname=PFCMemoria&path=%2F&sc=0>, implementa una Plantilla para la memoria de los proyectos de fin de carrera pertenecientes a la Facultad de Ingenieria de la Universidad de Deusto. No obstante la clase `memoriaPFC.cls` es a nuestro juicio algo pobre. Esta opinión se basa en el hecho de que dicha clase es una copia de la documentclass `scrbook`, a la cual se le ha añadido una macro para la inserción de figuras y dos entornos para formatear código fuente. Por

dicha causa no creemos que pueda ser considerada como un antecedente relevante de la clase `pclass.cls`, ya que ésta última ha sido creada desde cero, sin basarse en ninguna clase existente con anterioridad.

### Análisis temporal y costes de desarrollo

---

#### 6.1– Análisis temporal

En los apartados anteriores hemos descrito fielmente todo el proceso que ha dado como resultado la creación de la clase `LATEX pclass`. Una vez cubierta la descripción del proceso, pasaremos a tratar en detalle la planificación seguida para la realización de este proyecto, así como un desglose de las labores más importantes. Dicha tarea la desarrollaremos a lo largo del capítulo que nos ocupa en este instante.

Para comenzar con el análisis temporal, hemos de distinguir todas las etapas que nos han ocupado durante la elaboración de este proyecto. Un desglose de estas etapas sería:

1. **Aprendizaje y documentación sobre `LATEX`.**

Al comienzo del desarrollo la solución al problema propuesto, formateo de memorias para proyectos de fin de carrera, y su implementación usando lenguaje `LATEX` era completamente desconocida para ambos autores del proyecto. Para hacer esta afirmación nos basamos en que ninguno de los autores gozaba de conocimiento previos acerca de `LATEX`, por lo que la construcción de esta clase constituía nuestro primer contacto con dicho lenguaje de programación. Debido a lo anterior resultaba indispensable un periodo inicial de aprendizaje y asimilación de conceptos básicos sobre `LATEX`, así como una búsqueda extensa de documentación a la cual poder acudir en cualquier fase del proyecto.

2. **Creación de la clase `pclass.cls`**

Una vez hemos adquirido los conocimientos necesarios durante la fase anterior, en esta etapa procedemos a la creación en sí misma de la clase `pclass.cls`.

### 3. Fase de pruebas y testeo de pclass

Las constantes ideas que surgían para dar solución a los problemas que nos encontrábamos, dieron como resultado que la clase estuviese en todo momento sujeta a gran cantidad de cambios. Por esta razón resultaba imprescindible un periodo de pruebas y testeo, para así validar los nuevos elementos introducidos y su correcto funcionamiento con los elementos ya existentes.

Además una vez obtenido el contenido definitivo de `pclass.cls`, era necesario la creación de la plantilla para una memoria de proyecto, a la cual aplicaríamos el formato definido en el `.cls`. La finalidad de esta plantilla no es más que facilitar al usuario la redacción de su memoria, olvidándose del formato de la misma. Es decir, el usuario hará uso de la plantilla y la rellenará con los datos de su memoria, para finalmente obtener como resultado una memoria acorde al formato establecido para la presentación de la misma.

### 4. Elaboración de los capítulos de esta memoria

Durante esta fase hemos realizado la redacción de los capítulos que conforman la memoria de este proyecto.

### 5. Creación del estilo bibliográfico `pfcbibstyle`

Durante esta etapa hemos creado, haciendo uso de `Custom-bib`, un nuevo y propio estilo bibliográfico, el cual aplicaremos a las referencias de esta memoria. Dicho estilo recibe el nombre de `pfcbibstyle`. Hubiera resultado más sencillo emplear algunos de los estilos existentes más comunes. Sin embargo esta creación nos permitirá solucionar algunos problemas derivados del inglés como la ordenación errónea por orden alfabético de la letra ñe. Así mismo nos permitirá agrupar las referencias por orden alfabético en lugar de por orden de aparición como suele ser más habitual, además en las citas no aparecerá un número sino las iniciales del autor o autores y el año de publicación.

La planificación temporal consistirá en una estimación del tiempo dedicado a cada una de las fases de desarrollo del proyecto. Para ello asignaremos a cada una de esas fases las dos estimaciones siguientes:

1. **Estimación Inicial** la cual es empleada en los inicios del desarrollo. Suele ser poco exacta, pero se usan como primera aproximación para la viabilidad del proyecto.
2. **Estimación Final** la cual expresa la duración y el esfuerzo real empleado.

Para realizar una evaluación de la exactitud de la estimación se realiza una comparación de los valores reales con los valores estimados, teniendo

Tarea	Est. Inicial	Est. Final	RE
Documentación sobre L <sup>A</sup> T <sub>E</sub> X	25 días	36 días	30.5 %
Creación <code>pclass.cls</code>	18 días	19 días	5.3 %
Fase de pruebas y testeo	12 días	14 días	14.3 %
Elaboración capítulos memoria	8 días	10 días	20 %
Creación <code>pfcbibstyle</code>	5 días	5 días	0 %

Cuadro 6.1: Planificación del proyecto

en cuenta el error relativo medio. Tomando  $RE = \frac{A-E}{A}$ , donde A representa el valor real y E el valor estimado previamente, calculamos el error relativo medio mediante la expresión:

$$\left(\frac{1}{n}\right) \sum_{i=1}^n RE$$

Las estimaciones de cada fase se han realizado en días, considerando una dedicación de dos personas durante una media de 4 horas al día. Dicho esto obtendríamos las estimaciones mostradas en la Tabla 6.1. Analizando los datos de la misma obtenemos un error relativo medio de 14.02 %.

## 6.2– Costes de desarrollo

En esta sección estudiaremos detalladamente los costes que han supuesto la creación de la clase `pclass.cls`. Comenzaremos diciendo que no ha sido necesaria inversión alguna destinada a la adquisición de dispositivos específicos, hardware o software. Este hecho ha supuesto una gran ventaja a la hora de elaborar el proyecto que nos ocupa, ya que los costes se centrarán únicamente en los campos de costes de personal y costes indirectos.

### 6.2.1. Inversiones

En lo que respecta a la inversión para la adquisición de dispositivos específicos, tenemos que decir que en nuestro caso dicha inversión ha sido nula. Esto se debe a que no requeríamos de ningún material específico para materializar la clase `pclass`.

### 6.2.2. Costes de software

En lo que respecta los costes de software usado para el desarrollo del proyecto, hemos de indicar primeramente que todo el software empleado

es free software, o lo que es lo mismo software libre. Esto supondrá un coste cero en lo que respecta a este apartado.

Además del coste cero, el software libre también nos proporciona muchas otras ventajas. Este tipo de software brinda libertad a los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software; de modo más preciso, se refiere a cuatro libertades de los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo que puede ayudar a otros; de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie. El software libre suele estar disponible gratuitamente, o al precio de coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así, por ende no hay que asociar la idea de software libre a software gratuito.

Entre el software usado podemos citar entre otros:

- T<sub>E</sub>XLive, en caso de que usemos Linux.
- MikT<sub>E</sub>X, en caso de usar Windows.
- Kyle, como editor L<sup>A</sup>T<sub>E</sub>X sobre Linux.
- T<sub>E</sub>XnicCenter, como editor L<sup>A</sup>T<sub>E</sub>X sobre Windows.
- GhostScript y GSview, como visores de documentos.

En lo que respecta a la variedad de paquetes utilizados para complementar a la clase `pclass`, todos ellos son propuestas de software libre bajo licencia GNU, implicando también un coste nulo.

### 6.2.3. Costes de hardware

En el proceso de desarrollo ha sido necesaria la adquisición de dos equipos informáticos. Concretamente hemos hecho uso de dos PC portátiles Dell XPS 1530, los cuales hemos incluido como costes de hardware en el presupuesto. El valor que representan estos equipos es de 999 euros/unidad.

### 6.2.4. Costes indirectos

Dentro de esta sección dedicada a los costes indirectos incluiremos el gasto producido por material consumible (CD, impresora, papel, etc), luz consumida, limpieza de la zona de desarrollo, etc. Hemos considerado adecuado que un incremento del 5% en el presupuesto total sería representativo de este gasto.

Inversiones	0 euros
Costes Software	0 euros
Costes Hardware	1998 euros
Costes Personal	4388.16 euros
Costes Indirectos	319.31 euros
<b>TOTAL</b>	<b>6705.47 euros</b>

Cuadro 6.2: Presupuesto del proyecto

### 6.2.5. Costes de personal

Para calcular el coste personal hemos utilizado como referencia los sueldos brutos estipulados mínimos fijados en el BOE del 21 de Marzo de 2007, en el Anexo III. El sueldo anual para un Titulado de grado medio es de 14637.56 euros ó  $6.53 \frac{\text{euros}}{\text{hora}}$ .

También debemos tener en cuenta el número total de horas dedicadas a la elaboración de este proyecto. Dicho dato tiene un valor de 672 horas, representado por el trabajo realizado por dos personas durante 84 días dedicando una media de 4 horas/día.

De esta forma el coste personal supondrá una cuantía de:

$$672 \text{ horas} \cdot 6,53 \frac{\text{euros}}{\text{hora}} = 4388,16 \text{ euros.}$$

### 6.2.6. Presupuesto

Haciendo uso de los costes especificados anteriormente, obtendremos el presupuesto mostrado en la Tabla 6.2.



---

## Comparación con otras alternativas

---

A la hora de escribir la memoria de un proyecto de fin de carrera, la mayoría de usuarios recurre a los procesadores de texto más comunes (Word Perfect, Microsoft Word,...). Dichos procesadores son conocidos como procesadores de tipo pantalla o procesadores WYSIWYG, es decir, lo que ves es lo que obtienes. Sin embargo, existe una alternativa en el lenguaje  $\text{\LaTeX}$ , el cual nos proporcionará múltiples ventajas. A través de la utilización de la clase `pclass` intentaremos que el usuario perciba estas innumerables ventajas.

Para compara ambas alternativas describiremos las distintas etapas que componen la elaboración de un texto en  $\text{\LaTeX}$  y en los procesadores de pantalla. Estas etapas deberán ser completadas por el usuario partiendo de una primera versión del documento en borrador, hasta llegar a la salida por la impresora del documento definitivo.

De este modo durante la elaboración del documento con  $\text{\LaTeX}$  podemos distinguir tres etapas:

### 1. **Etapa 1: Preparación**

Consiste en la escritura del documento a procesar mediante un editor de textos. Una vez finalizada esta etapa el documento podrá ser tratado por el procesador.

### 2. **Etapa 2: Procesado**

Una vez que tenemos el texto escrito, éste servirá de fichero fuente para el procesador. El procesador interpretará las órdenes escritas en él y convertirá el texto en la llamada composición.

En caso de que el texto fuente contenga errores de programación, el procesador los localizará y posiblemente no obtengamos una composición con los resultados deseados. Si estamos en este caso deberemos volver a la etapa 1, corregir tales errores y continuar.

### 3. Etapa 3: Impresión

Una vez tenemos la composición adecuada, ésta será enviada a una impresora, obteniendo así la versión final del documento sobre el papel. Logicamente si la salida resulta impropia (faltas de ortografía, invasión de márgenes,...) deberemos volver nuevamente a la etapa 1 para subsanar los errores ocurridos.

Una vez descritas las etapas de creación de un documento  $\text{\LaTeX}$ , podemos observar que estas difieren notablemente de las que se realizan con los procesadores de tipo pantalla.

En los procesadores de texto de pantalla la edición y la composición del documento se realizan al mismo tiempo, no existiendo distinción entre una y otra. Durante la edición la pantalla mostrará constantemente el resultado de la composición. Por ejemplo si queremos escribir la letra A, bastará con pulsar la tecla correspondiente a dicho carácter. Pero si en cambio queremos que dicho carácter aparezca en negrita y a veinte puntos podremos hacerlo, y sobre la pantalla dicho carácter aparecerá en negrita y a 20 puntos.

De esta manera sólo podemos distinguir dos etapas bien diferenciadas en la creación de un documento con un procesador de pantalla:

#### 1. Etapa 1: Edición/Composición

La composición del documento se realiza de forma directa e interactiva. En todo momento, la pantalla muestra el estado actual de la misma, tal y como la veríamos impresa sobre el papel.

#### 2. Etapa 2: Impresión

Finalizada la etapa anterior, es deseable tener una versión sobre el papel. Esta etapa coincide con la tercera etapa de la creación de un documento con  $\text{\LaTeX}$ .

Podemos observar que, a diferencia de  $\text{\LaTeX}$ , aquí no existe el concepto de código fuente sobre el que actuará posteriormente el procesador. Para usuarios habituales de procesadores de pantalla, el hecho de que se muestre el estado actual de la composición durante la edición supone una enorme ventaja sobre  $\text{\LaTeX}$ .

Dicho esto todos nos preguntamos cuales son las ventajas de  $\text{\LaTeX}$  sobre los procesadores de pantalla. La respuesta la podemos encontrar en el mismo nombre WYSIWYG, pero debiendo aclarar que lo que ves es lo que obtienes y nada más. Ahora mostrar la composición en pantalla durante la edición se torna en un inconveniente: dificultado para realizar cambios globales en el documento, al modificar la numeración en un punto del escrito (por ejemplo, introduciendo un capítulo nuevo o sección) deberemos modificar a mano el resto de la numeración, etc. Por el contrario, la composición con  $\text{\LaTeX}$  no representa la rigidez de los procesadores de

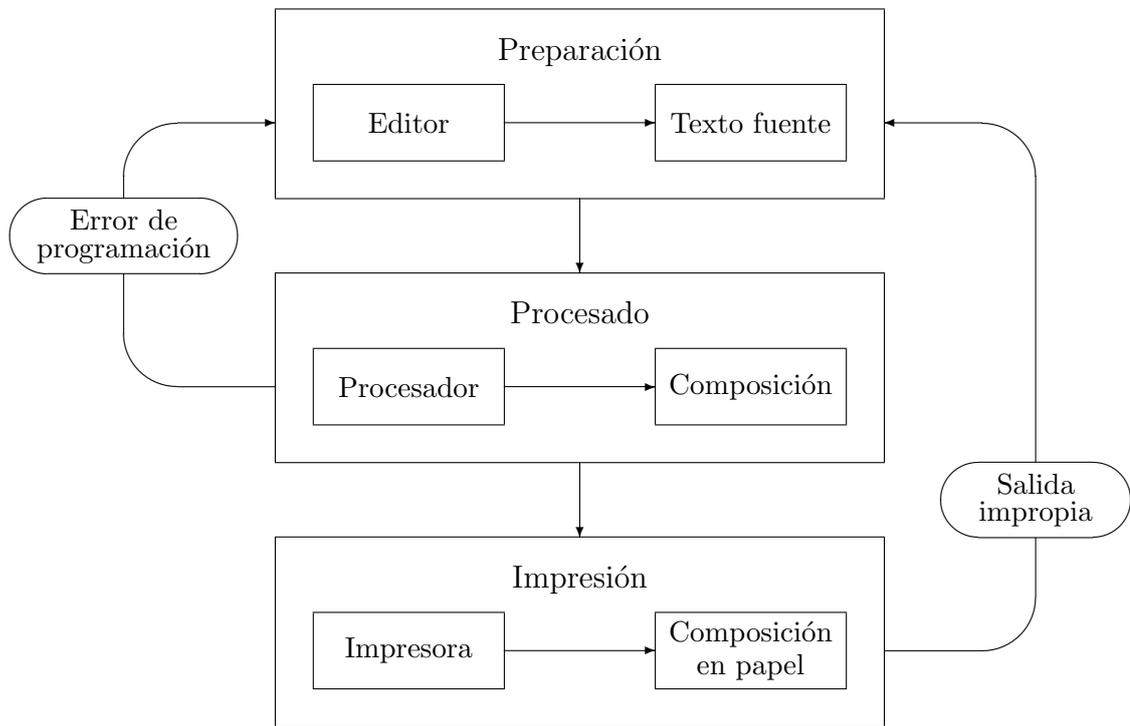


Figura 7.1: Etapas de creación de un documento  $\text{\LaTeX}$

pantalla, esto se debe a que ésta se efectúa a partir de las instrucciones contenidas en el fichero fuente.

Por otro lado debemos añadir que  $\text{\LaTeX}$  se comporta como un lenguaje de programación y componedor al mismo tiempo, lo que le confiere unas posibilidades que ningún otro procesador posee, la calidad de composición de los documentos elaborados bajo sus órdenes así lo avalan.



### Pruebas y conclusiones

---

#### 8.1– Pruebas

Este apartado de pruebas será bastante breve. Este hecho se debe a que la clase  $\text{\LaTeX}$  `pclass.cls` que da origen a este proyecto de fin de carrera, tiene como finalidad obtener un formato adecuado para la memoria de la asignatura proyecto informático de las titulaciones: Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas. Es evidente que el formato definido obedecerá al reglamento establecido por la Universidad de Sevilla para dicha asignatura.

Como consecuencia de lo expresado en el párrafo anterior, qué mejor prueba puede existir que presentar la documentación correspondiente a la creación de la clase `pclass.cls`, que aplicar a este documento el formato definido en esta misma clase. De esta forma la documentación que usted tiene en sus manos en este mismo instante, constituye la mejor y única prueba de el proyecto que da lugar a esta memoria.

#### 8.2– Conclusiones

Finalmente y como colofón a esta memoria, describiremos en este apartado las distintas conclusiones que hemos obtenido durante la elaboración de este proyecto. Debemos recordar que en dicho proyecto hemos creado una clase  $\text{\LaTeX}$ , llamada `pclass.cls`, cuya meta era el formateo de memorias de la asignatura proyecto de final de carrera, para las diversas titulaciones impartidas en la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla.

Para comenzar, queremos destacar las dificultades que surgen a todo usuario una vez que emprende la creación de una clase  $\text{\LaTeX}$ . Cuando hablamos de dificultades nos estamos refiriendo principalmente al amplio

periodo inicial de aprendizaje que requiere este lenguaje. Esta etapa inicial, que puede resultar bastante tediosa, es imprescindible para poder conseguir de forma exitosa la creación de tu propia clase `.cls`. Este factor se acentúa de manera notable si el usuario no posee previamente una serie de conceptos básicos acerca del lenguaje  $\text{\LaTeX}$ . Este hecho lo podemos corroborar los autores de este proyecto, ya que para ambos la creación de la clase `pclass.cls` supone nuestro primer contacto con el mundo  $\text{\LaTeX}$ .

Pero no todo lo relacionado con  $\text{\LaTeX}$  va a resultar negativo, una vez superado el extenso periodo descrito en el párrafo anterior, todas las experiencias venideras resultarán altamente gratificantes. Un ejemplo de ello son la multitud de formatos que ofrece este lenguaje, permitiéndote haciendo uso de ellos cambiar de arriba a abajo la apariencia de un documento de manera casi instantánea. Por esta entre otras razones  $\text{\LaTeX}$  es considerado por muchos como un lenguaje muy potente.

En lo que respecta a la clase `pclass` hemos percibido, una vez hemos finalizado con su creación, que dicha clase puede establecerse como una herramienta de gran ayuda para cualquier estudiante que se encuentre inmerso en la elaboración de su proyecto de final de carrera. Para realizar la afirmación anterior, nos remitimos a las innumerables ventajas que puede suponer elaborar la documentación de un proyecto haciendo uso de plantilla proporcionada con `pclass.cls`. Entre ellas podríamos citar, entre otras, la generación automática de los Índices General, de Tablas y Figuras, gestión automática de las referencias bibliográficas, generación de la portada según el formato establecido,...

También queremos reflejar la visión global de los autores de este proyecto acerca de la clase `pclass`. Creemos que esta clase puede ser tremendamente útil para cualquier estudiante, incluso si éste no posee conocimiento alguno acerca de  $\text{\LaTeX}$ , en cuyo caso le será de gran ayuda el Capítulo 11. A través de `pclass` se obtendrá una documentación con un formato más que aceptable. Todo ello teniendo en cuenta como hemos dicho que  $\text{\LaTeX}$  es un lenguaje muy potente por lo cual siempre existirán posibles mejoras futuras. Por este motivo siempre estaremos dispuestos a recibir sugerencias que supongan futuras mejoras para `pclass.cls`.

Una vez analizado el contenido anterior en este capítulo, comprenderás, al igual que nosotros, que los inconvenientes que citábamos sobre  $\text{\LaTeX}$  no son más que una minucia si los comparamos con los espectaculares resultados que nos proporciona dicho lenguaje. Así que sólo podemos terminar animando a todos a zambullirse en el apasionante universo  $\text{\LaTeX}$ .

## CAPÍTULO 9

---

### Clase proyecto fin de carrera pclass

---

#### 9.1— Introducción

Lo primero que haremos cuando querramos poner comandos nuevos en un archivo  $\text{\LaTeX}$  es decidir si eso será *documento clase* o *paquete*. La regla para esto es simple:

Si los comandos se pueden usar con cualquier clase, entonces será un paquete, pero sin embargo si no se puede hacer eso será una clase.

Hay dos tipos principales de clase: tales como `article`, `report` o `letter`, los cuales son estándares; y otros que son extensiones o variaciones de estos últimos, por ejemplo, el documento de clase `proc` el cual esta construido a partir de un documento clase `article`.

#### 9.2— Estructura de la clase

Las clases en  $\text{\LaTeX} 2_{\epsilon}$  tiene más estructura de la que el estilo  $\text{\LaTeX} 2.09$  tenía. El esbozo de una clase o un paquete de archivos es:

**Identificación** El archivo dice si el mismo es un paquete de  $\text{\LaTeX} 2_{\epsilon}$  o si es una clase, y da una breve descripción de sí mismo.

**Declaraciones preliminares** Aquí el archivo declara algunos comandos que pueden ser cargados por otro archivos. Esos comandos suelen ser necesitados por el código utilizado en la declaración de opciones

**Opciones** El archivo declara y procesa estas opciones.

**Más declaraciones** Aquí es donde el archivo hace la mayoría del trabajo: declarando nuevas variables, comandos y fuentes, y cargando otros archivos.

### 9.2.1. Identificación

Lo primero que hace una clase o un paquete es identificarse a si misma, y lo hace de la manera que sigue abajo:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{<paquete>}[<fecha> <otra información>]
```

Por ejemplo:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{latexsym}[1994/06/01 Std. LaTeX package]
```

Las clases lo hacen así:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{<class-name>}[<fecha> <otra información>]
```

Por ejemplo:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{article}[1994/06/01 Standard LaTeX class]
```

La *<fecha>* se debe dar de la forma ‘YYYY/MM/DD’ y debe de estar presente siempre que el argumento opcional es usado (esto se cumple para el comando `\NeedsTeXFormat`). Cualquier cambio en esta sintaxis supondrá un error de bajo-nivel en  $\TeX$ , el comando espera una sintaxis válida para usar el paquete o la clase y no está previsto de solución para este tipo de errores. Esta fecha está marcada cada vez que un usuario especifica una fecha en su comando `\documentclass` or `\usepackage`. Por ejemplo, si escribes:

```
\documentclass{article}[1995/12/23]
```

los usuarios de otra localización diferente podrían obtener un aviso de que su copia de `articulo` esta caducada.

La descripción de una clase aparece cuando la clase se utiliza. La descripción de un paquete se coloca en el archivo de registro. Estas descripciones también se representan por el comando `\listfiles`. La frase **Standard LaTeX** **no debe** ser usada en la identificación de cualquier archivo que no sea un estándar de la distribución de  $\LaTeX$ .

### 9.2.2. Usando una clase

La mayor diferencia entre los archivos de estilo de  $\LaTeX$  2.09 y las clases y paquetes de  $\LaTeX$  2 $\epsilon$  es que  $\LaTeX$  2 $\epsilon$  soporta la *modularidad*, en el sentido de crear archivos de pequeña construcción de bloques en lugar de utilizar solo los archivos grandes. Un paquete o clase  $\LaTeX$  puede cargar paquetes como sigue:

```
\RequirePackage[<options>]{<package>}[<date>]
```

Por ejemplo:

```
\RequirePackage{ifthen}[1994/06/01]
```

Este comando tiene la misma sintaxis que el comando `\usepackage`. Este permite a los paquetes o clases usar las características proporcionadas por otros paquetes. Por ejemplo, cargando el paquete `ifthen`, un programador de paquetes puede usar los comandos ‘`if...then...else...`’ proporcionados por el paquete.

Una clase  $\text{\LaTeX}$  puede cargar a otra clase tal como sigue:

```
\LoadClass[<options>]{<class-name>}[<date>]
```

Por ejemplo:

```
\LoadClass[twocolumn]{article}
```

Este comando tiene la misma sintaxis que el comando `\documentclass`. Esto permite a las clases basarse en la sintaxis o en la apariencia de otra clase. Por ejemplo, cargando la clase `article`, a un programador que no le gusta la clase `article` solo tiene que cambiar los bits en lugar de escribir una nueva.

Los siguientes comandos pueden ser utilizados en el caso común que se desea simplemente cargar una clase o un paquete de archivos con exactamente esas opciones que están siendo utilizados por la clase actual.

```
\LoadClassWithOptions{<class-name>}[<date>]
\RequirePackageWithOptions{<package>}[<date>]
```

Por ejemplo:

```
\LoadClassWithOptions{article}
\RequirePackageWithOptions{graphics}[1995/12/01]
```

### 9.2.3. Declarando opciones

Otra de las diferencias entre los estilos de las clases en  $\text{\LaTeX}$  2.09 y  $\text{\LaTeX}$  2<sub>ε</sub> son las opciones de manejo. Los paquetes y las clases pueden ahora declarar opciones y estas pueden ser especificadas por los programadores; por ejemplo, la opción `twocolumn` se declara por la clase `article`. Tenga en cuenta que el nombre de una opción debe contener solamente los caracteres permitidos en un nombre  $\text{\LaTeX}$ ; en particular no deben de contener ninguna secuencia de control.

Una opción se declara como sigue:

```
\DeclareOption{<option>}{<code>}
```

Por ejemplo, la opción `dvips` (algo simplificada) a el paquete `graphics` se implementa así:

```
\DeclareOption{dvips}{\input{dvips.def}}
```

Esto significa que cuando un programador escriba.

`\usepackage[dvips]{graphics}`, el archivo `dvips.def` se carga. Como otro ejemplo, la opción `a4paper` se declara en la clase `article` para configurar los parámetros `\paperheight` y `\paperwidth`:

```
\DeclareOption{a4paper}{%
  \setlength{\paperheight}{297mm}%
  \setlength{\paperwidth}{210mm}%
}
```

A veces un usuario requerirá una opción la cual es no esta explícitamente declarada en las opciones de la clase o paquete. Por defecto esto producirá un *warning* en una clase o un *error* en un paquete; este comportamiento se puede alterar de la siguiente manera:

```
\DeclareOption*{<code>}
```

Por ejemplo, para que el paquete `fred` produzca un error de opciones desconocida, podemos poner:

```
\DeclareOption*{%
  \PackageWarning{fred}{Unknown option ‘\CurrentOption’}%
}
```

Luego, si el programador escribe `\usepackage[foo]{fred}`, obtendrán un *warning* `Package fred Warning: Unknown option ‘foo’..` Como otro ejemplo, el paquete `fontenc` intenta cargar el archivo `<ENC>enc.def` siempre que sea `<ENC>` la opción usada. Ester archivo puede ser escrito así:

```
\DeclareOption*{%
  \input{\CurrentOption enc.def}%
}
```

Es posible pasar opciones a otros paquetes o clases, usando el comando `\PassOptionsToPackage` o `\PassOptionsToClass` (observar que esta operación esta especializada y solo trabaja para nombres de opciones). Por ejemplo, para pasar cada opción desconocida a la clase `article`, se puede usar:

```
\DeclareOption*{%
  \PassOptionsToClass{\CurrentOption}{article}%
}
```

Si haces esto te asegurarás de cargar la clase en algún punto más tarde, de otra manera las opciones nunca serán procesadas.

Hasta ahora, hemos explicado sólo como se declaran las opciones, no cómo ejecutarlos. Para procesar las opciones con que el archivo se llama, se debe usar:

```
\ProcessOptions\relax
```

Esto ejecuta el  $\langle code \rangle$  para cada opción .

Por ejemplo, si el paquete `jane` contiene el archivo:

```
\DeclareOption{foo}{\typeout{Saw foo.}}
\DeclareOption{baz}{\typeout{Saw baz.}}
\DeclareOption*{\typeout{What's \CurrentOption?}}
\ProcessOptions\relax
```

y el programador escribe `\usepackage[foo,bar]{jane}`, obtendrá el mensaje `Saw foo. y What's bar?`.

#### 9.2.4. Clase mínima de ejemplo

La mayoría del trabajo en un clase es definir nuevos comandos, o cambiar la apariencia de los documentos. Esto se hace en el cuerpo de los paquetes, usando comandos tales como `\newcommand` o `\setlength`.

$\text{\LaTeX} 2_{\epsilon}$  tiene varios comandos nuevos para ayudar a los programadores de clases y paquetes. Hay cuatro cosas que cada archivo clase *debe* contener: estas son una definición de `\normalsize`, valores para `\textwidth` y `\textheight` y una especificación para la numeración de páginas. Así que un documento de clase mínimo <sup>1</sup> queda como sigue:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Std LaTeX minimal class]
\renewcommand{\normalsize}{\fontsize{10pt}{12pt}\selectfont}
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{8in}
\pagenumbering{arabic}
```

Sin embargo, este archivo clase no soportará `\footnotes`, `\marginals`, `\floats`, etc., y tampoco un tipo 2-letter o fuentes y comandos tales como `\rm`; todo esto debe de contener algo más que esta clase mínima

<sup>1</sup>Esta clase esta ahora en la distribución estándar, como `minimal.cls`

### 9.2.5. Código de opciones clase pclass

En nuestra clase las opciones que hemos tomado han sido sencillas, sin cargar clases con opciones o paquetes con opciones de otros archivos. En la clase nos basamos en el estándar de L<sup>A</sup>T<sub>E</sub>X book, y a partir de este modificaciones tanto de configuración de página como diferentes macros para facilitar el trabajo. Como se puede observar en la Captura. A continuación tenemos la declaración de opciones en la clase pclass.

```

1
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesClass{pclass}
4 [2008/09/08 PFC class Universidad de Sevilla]
5
6 \DeclareOption{openright}{ \PassOptionsToClass{openright}{book} }
7 \DeclareOption{openany}{ \PassOptionsToClass{openany}{book} }
8 \DeclareOption{twoside}{ \PassOptionsToClass{twoside}{book} }
9 \DeclareOption{oneside}{ \PassOptionsToClass{oneside}{book} }
10 \ExecuteOptions{twoside,openright}
11 \ProcessOptions
12 \LoadClass[a4paper,11pt]{book}
13
14 \textwidth=350pt
15 \textheight=600pt
16 \marginparwidth=100pt

```

Código 9.1: Código de opciones de clase pclass

## 9.3– Clase pclass

A pesar de que una clase define el amplio documento en los términos de composición tipográfica, L<sup>A</sup>T<sub>E</sub>X necesita instrucciones tipográficas adicionales para dar formato a un documento completo. Algunas de estas instrucciones provienen de la opciones clase documento, una colección de formato de instrucciones que definen tipográfica con más detalle. Las opciones que puede controlar el texto en el tamaño de la letra, la orientación de la página, el número de columnas de texto, calidad de impresión, tamaño de página, y muchos otros aspectos del documento de diseño y composición tipográfica. En el siguiente cuadro tenemos un resumen de las opciones por defecto de la clase book.cls y los posibles cambios que podemos hacerle.

Un documento L<sup>A</sup>T<sub>E</sub>X puede dividirse en varias partes, en concreto la clase book en la cual nos hemos basado permite dividir el documento en partes, cada una de ellas obtenidas con el comando `\part{}`. Con la clase book podemos incluir también capítulos. Para iniciar un capítulo usamos el comando `\chapter{título}`. Similarmente se usan los comandos:

`\section{}`, `\subsection{}`, `\paragraph{}` y `\subparagraph{}`.

Categoría	Por defecto	Opciones
Tamaño en puntos del cuerpo	10 pt	11 pt, 12 pt
Tamaño del pael	a4	a5, b5
Orientación	Portrait	Landscape
Lados para imprimir	Both sides	One side
Calidad	Final	Draft
Título de página	Title page	No title page
Columnas	One	Two
Empezar el capítulo en la izquierda	No	Yes
Ecuación de numeración	On right	On left
Ecuaciones mostradas	Centered	Flush left
Estilo para abrir bibliografía	Compressed	Open

Cuadro 9.1: Opciones para book.cls

Para secciones, subsecciones, etc. Cabe mencionar que la numeración de cada una de estas partes del documento la realiza  $\text{\LaTeX}$  por sí solo. Hay ocasiones en que el título de una sección es muy largo. En estos casos, el encabezado con el nombre de la sección sobrepasa el tamaño de la página, por lo que es conveniente contar con un método para que el nombre de la sección aparezca abreviado en el encabezado de la página. Por ello, la forma general del comando para secciones que provee  $\text{\LaTeX}$  es el siguiente:

```
\section[nombre corto de la sección]{nombre de la sección}
```

El cuadro 9.2 muestra las distintas unidades disponibles para las clases book, así como los comandos necesarios para declararlos:

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

Cuadro 9.2: Jerarquía de las unidades de estructura según la clase book

Además, esta clase de documento incluye nuevas características en  $\text{\LaTeX 2}_\epsilon$ , como lo son los comandos que declaran grandes unidades de estructuras de un libro.:

```
\frontmatter (título, prefacios, introducción, etc...)
\mainmatter (parte principal del libro)
```

`\backmatter` (índices de materias, alfabéticos, etc...)

Con `\frontmatter` damos el estilo que debe tener la parte frontal del libro (página de título, tabla de contenidos, prólogos), con `\mainmatter` damos el estilo que debe tener el texto principal del documento, y finalmente `\backmatter` se usa para el estilo de la parte final del libro (la bibliografía, los índices de materias).

Todo lo que quede contenido entre `\frontmatter` y `\mainmatter` (que se supone debe de ser la parte frontal del libro), tendrá un estilo en el que la numeración de página es con números romanos, y ningún capítulo, ni ningún otro título de nivel inferior, será numerado. Las páginas después de `\mainmatter` serán numeradas con números arábigos y los capítulos y títulos de nivel inferior sí serán numerados. Con `\backmatter` hacemos que los capítulos y títulos nivel inferior no aparezcan numerados (lo que es ideal para conclusiones o notas finales).

### 9.3.1. Frontmatter

También se conoce como preámbulo, en nuestra clase es la parte que contiene la portada del proyecto, los agradecimientos, un resumen del mismo, el índice general, el índice de tablas, índice de figuras e índice de código. La numeración será en románico y empezando des de "I" a partir de la portada. Tanto los agradecimientos como el resumen no tiene encabezado de título. El código de `fronttmatter` es el siguiente.

```

1  \frontmatter
2  % Introduce como resumen el contenido de resumen.tex
3  \input{resumen.tex}
4  % Introduce como agradecimientos
5  % el contenido de agradecimientos.tex
6  \input{agradecimientos.tex}
7  %Indice de contenidos
8  \tableofcontents
9  % Indice de tablas
10 \listoftables
11 % Indice de figuras
12 \listoffigures
13 % Indice de capturas de codigo
14 \lstlistoflistings

```

Código 9.2: Frontmatter en proyect.tex

Para los índices de figuras e índices de código hemos modificado las definiciones para que la salida sea `tabla` en vez de `cuadro` y `código` en vez de `code`.

```

1 \addto\captionsspanish{
2   \renewcommand{\tablename}{Tabla}}
3 \addto\captionsspanish{
4   \renewcommand{\listtablename}{Indice de tablas}}
5
6 \renewcommand{\lstlistingname}{Indice de codigo}
7 \renewcommand{\lstlistingname}{Codigo}

```

Código 9.3: Modificación de tabla y código

En el Índice general podemos definir hasta que nivel queremos incluir elementos, en nuestro caso solo tenemos hasta subsección, nivel 2:

```

1 \setcounter{tocdepth}{2}

```

Código 9.4: Nivel de profundidad

## Portada

La portada por defecto podemos verla en <http://www.informatica.us.es/docs/portada.pdf>, en nuestra clase tenemos definidas macros que se pueden ver en 9.6 con las cuales solo tenemos que incluir los datos y hacer una llamada para la la creación de la portada. El código de llamada es el siguiente.

```

1 \newcommand{\hacerportada}{
2   \begin{titlepage}
3     \centering
4     \includegraphics[scale=2.5]{img/us}
5     \begin{center}\bf\sffamily
6       {\normalsize ESCUELA TECNICA SUPERIOR DE INGENIERIA INFORMATICA
7         }\\[1cm]
8       {\large \@titulacion{}}\\[2.5cm]
9       {\large \@titulopro{}}\\[2cm]
10      {\large Realizado por}\\
11      {\large \@autorUno{}}\\
12      {\large \@autorDos{}}\\
13      [1cm]
14      {\large Dirigido por}\\
15      {\large \@tutor}\\[1cm]
16      {\large Departamento}\\
17      {\large \@departamento}\\[3cm]
18    \end{center}
19    \begin{flushright}
20      {\bf\sffamily\large Sevilla, ({\large \@diad})}
21    \end{flushright}
22  }

```

Código 9.5: Creación de portada

### 9.3.2. Mainmatter

Es la parte que contiene los capítulos principales de los que se componen el proyecto, empieza justo después de los listados de índices y desde aquí se comienza la numeración a partir del "1" en números arábigos.

```

1  \mainmatter
2      % inician los capitulos de tu proyecto
3      % con la numeracion arabic
4
5      %\include{capitulo1} %es equivalente a
6      %\clearpage \input{file} \clearpage
7      \input{Capitulos/capitulo1}
8      \input{Capitulos/capitulo2}
9      \input{Capitulos/defobjetivos}
10     \input{Capitulos/requisitos}
11     \input{Capitulos/analanteced}
12     \input{Capitulos/analttemporal}
13     \input{Capitulos/alternativas}
14     \input{Capitulos/pruebas}
15     \input{Capitulos/capitulo3}
16     \input{Capitulos/capitulo4}
17     \input{Capitulos/manual}

```

Código 9.6: Mainmatter en proyect.tex

El documento será a dos caras y siempre los capítulos empezarán por una página derecha, esto se consigue en la clase de esta forma:

```

1  \ExecuteOptions{twoside,openright}

```

Código 9.7: ExecuteOptions en pclass.cls

Para todas las páginas izquierdas que no contienen información se deben eliminar las cabeceras los títulos de páginas y la numeración, esto se conseguía antes con una macro algo complicada pero como utilizamos el paquete `titlesec` para configurar todas estas opciones podemos también utilizarlo para que haga la misma función que la antigua macro y nos deje en blanco aquellas páginas sin información, y esto lo hacemos de la siguiente forma:

```

1  \usepackage[clearempty,pagestyles,newparttoc]{titlesec}

```

Código 9.8: Páginas sin información en blanco

Las dimensiones de las páginas las podemos ver en la subsección 9.4.2, las definiciones de tamaño en nuestra clase como las posibles opciones de cambio y una pequeña explicación la podemos ver a continuación en el

siguiente código:

```

1  %Dimension de paginas
2
3  \textwidth=350pt
4  \textheight=600pt
5  \marginparwidth=100pt
6
7  %\setlength{\textwidth}{x} %Ancho del cuerpo
8
9  %\setlength{\oddsidemargin}{x}
10     %1in + oddsidemargin = 1.54cm + x
11 %\setlength{\evensidemargin}{x}
12     %1in + evensidemargin = 1.54cm + x
13
14 %\setlength{\textheight}{x} %Alto del Cuerpo
15
16 %\setlength{\topmargin}{x}
17     %Distancia entre el margen superior y el encabezamiento /
18     headheight
19
20 %\setlength{\headsep}{x}
21     %Distancia entre el encabezamiento y el cuerpo
22
23 %\setlength{\footnotesep}{x}
24     %El alto de una estructura colocada al principio de cada
25     %footnote para crear un espacio vertical entre los footnotes

```

Código 9.9: Dimensión de páginas

En esta misma subsección tenemos la configuración y opciones de los títulos de cada página par e impar, la numeración y la forma en que aparecen las secciones y las subsecciones. También en la subsección 9.4.3 tenemos la configuración para los títulos de los capítulos.

Para itemización de los objetos podemos definir el formato de los símbolos a utilizar para cada item y su orden de aparición según en la profundidad en la que se encuentren, vemos un ejemplo.

- punto1
  - círculo
    - \* asterisco
    - ★ estrella
- punto2

Para hacer posible este formato del entorno `itemize` has sido necesario realizar las siguientes redefiniciones dentro de la clase.

```

1 %%%%%%%%%%%%%%REDEFINICION ITEMIZE%%%%%%%%%%%%%
2 \renewcommand{\labelitemi}{\bullet}%enumera con punto negro
3 \renewcommand{\labelitemii}{\circ}%enumera con circulo
4 \renewcommand{\labelitemiii}{\ast}%enumera con asterisco
5 \renewcommand{\labelitemiv}{\star}%enumera con estrella
6 %%%%%%%%%%%%%%

```

Código 9.10: Redefinición itemize

### 9.3.3. Backmatter

Es la sección del documento que se reservada usualmente para los apéndices y bibliografía. Es decir tras el comando `\backmatter` todos los input que se realicen propiciarán la aparición de un nuevo apéndice en tu memoria de proyecto.

Los apéndices aparecen al final del documento y además figuran en el Índice general sin continuar la numeración de los capítulos, aunque sus páginas siguen la numeración a continuación de éstos.

Es usual que el apartado referente a bibliografía constituya un apéndice dentro del documento. Para que la bibliografía sea creada a modo de apéndice es necesario realizar las declaraciones en el archivo `proyect.tex` de la siguiente manera:

```

1
2 %comienza el backmatter
3 \backmatter
4 % apendiice relativo a la licencia
5 \input{Capitulos/licencia}
6
7 % para generar la bibliografia
8 \bibliographystyle{pfc}
9 \bibliography{pfcbib}
10
11 \end{document}

```

Código 9.11: Backmatter

Para un estilo de bibliografía propio como el que aplicamos en la clase tenemos en la sección 9.5.

## 9.4– Paquetes

### 9.4.1. ¿Cómo y dónde instalo nuevos paquetes?

Este proceso constará en general de los pasos que se describen a continuación: traer el nuevo paquete, extraer los ficheros de estilo si es necesario, colocarlos en el sitio apropiado y rehacer la base de datos.

### Dónde buscar un paquete nuevo y qué traer

Normalmente los paquetes nuevos se encontrarán en el capítulo 3 (CTAN), aunque en ocasiones estarán en otros lugares. En general, se debe descargar el directorio completo del paquete o el archivo comprimido que lo contiene. Esto no es necesario cuando se quiere descargar un archivo de estilo de uno de los directorios *misc*, que tienen contribuciones al *CTAN* en forma de archivos de estilo individuales completos en sí mismos. En este caso bastaría con descargar el archivo individual correspondiente.

### ¿Qué es cada uno de los archivos que traigo?

Un paquete pequeño puede estar compuesto únicamente de un archivo de estilo `.sty` (por ejemplo `paquete.sty`) con las instrucciones de uso incluidas como comentarios en el mismo, en un archivo separado o bien en un archivo *README*.

Sin embargo, es más frecuente encontrar el paquete en forma de un par de archivos `paquete.ins` y `paquete.dtx`, escritos para ser utilizados con el sistema *doc* de  $\text{\LaTeX}$ . Los archivos de estilo deben extraerse de éstos. Si hay un *README* adicional debe leerse éste previamente.

### Extrayendo archivos de estilo de los `.dtx` y `.ins`

En el sistema *doc* el manual de usuario y el código del paquete documentado se encuentran en el archivo `.dtx`, mientras que el archivo `.ins` contiene instrucciones  $\text{\LaTeX}$  acerca de la extracción del código del archivo `.dtx`. Para extraer los distintos archivos debe seguirse el siguiente procedimiento:

- Correr  $\text{\LaTeX}$  sobre `paquete.ins`. Esto extraerá uno o más archivos (normalmente un `paquete.sty`, pero dependiendo del paquete pueden generarse más archivos).
- Correr  $\text{\LaTeX}$  sobre `paquete.dtx` para obtener el manual de usuario y posiblemente una versión comentada del código del paquete.
- Correr de nuevo  $\text{\LaTeX}$  sobre `paquete.dtx`. Ésto resolverá las referencias y generará una tabla de materias si el archivo original lo pide así.
- Si  $\text{\LaTeX}$  da el error `"No file paquete.ind"` significa que no encontró el archivo fuente para el índice de órdenes. Para generar el índice basta hacer

```
makeindex -s ind.ist paquete
```

y correr de nuevo  $\text{\LaTeX}$ .

- Imprimir y leer `paquete.dvi`.

A veces se proporciona el manual de usuario separadamente del archivo `.dtx`. En este caso es recomendable procesarlo después de hacer lo anterior, ya que puede necesitar elementos del paquete que está describiendo.

### ¿Dónde colocar nuevos archivos de estilo?

En primer lugar  $\text{\TeX}$  buscará archivos en el directorio actual. Salvo que se trate de una prueba o de archivos muy relacionados con el documento que se está preparando, es conveniente colocarlos en un lugar de acceso más general.

El lugar exacto en el que deben colocarse los nuevos archivos de estilo depende de la distribución  $\text{\TeX}$  que se esté utilizando. Asumiendo que se utiliza una de las distribuciones modernas que son conformes al TDS (por ejemplo,  $\text{teTeX}$ ,  $\text{fpTeX}$  o  $\text{mikTeX}$ ) hay una serie de normas que deben tenerse en cuenta

1. Instalar siempre los nuevos archivos personales en una rama *texmf* local del árbol global o en una rama personal, dependiendo de si son archivos para uso común en la máquina o únicamente para el usuario. De esta forma puede actualizarse el árbol *oficial* sin tocar los archivos locales o personales. Para la rama local, el directorio raíz local tendrá un nombre del tipo:

```
teTeX:      /usr/share/texmf.local/
fpTeX:      c:\fptex\texmf.local\
mikTeX:     c:\localtexmf\
```

que puede cambiar dependiendo de las opciones dadas durante la instalación. Por simplicidad en lo que sigue le denominaremos: `$TEXMFLOCAL`.

2. En la rama local, reproducir la estructura de directorios de la rama principal. Estos son unos ejemplos de dónde deberían colocarse archivos de distintas extensiones:

```
.sty,
.cls: $TEXMFLOCAL/tex/latex/<paquete>/
.dvi,
.ps,
.pdf: $TEXMFLOCAL/doc/latex/<paquete>/
.bib: $TEXMFLOCAL/doc/bibtex/bib
.bst: $TEXMFLOCAL/doc/bibtex/bst
.tfm: $TEXMFLOCAL/fonts/tfm/<suministrador>/<fuente>/
.vf:  $TEXMFLOCAL/fonts/vf/<suministrador>/<fuente>/
```

```
.afm: $TEXMFLOCAL/fonts/afm/<suministrador>/<fuente>/
.pfb: $TEXMFLOCAL/fonts/type1/<suministrador>/<fuente>/
.ttf: $TEXMFLOCAL/fonts/truetype/<suministrador>/<fuente>/
```

donde *paquete*, *fuentes* y *suministrador* dependen de cada archivo individual de cada paquete. La rama personal suele estar en un subdirectorío `texmf` del directorío de usuario, pero puede cambiar. En ella también es necesario reproducir la estructura de directoríos de la rama principal. Dependiendo de la distribución y/o de las opciones de configuración puede ser necesario rehacer la base de datos cuando se añaden o quitan elementos.

### Activando ramas locales y personales del árbol de directoríos $\LaTeX$ global

A menudo la rama local del árbol global no está activada por omisión y es necesario activarla:

#### *teTeX* y *fpTeX*;

En primer lugar es necesario localizar el archivo de configuración `texmf.conf`. Éste puede estar en `/etc/texmf/texmf.conf`, también en `/etc/texmf.conf` o `/usr/share/texmf/web2c/texmf.conf`, dependiendo de la distribución. Leer el principio del fichero, ya que puede haber sido generado automáticamente. Si es así, seguir las instrucciones que allí aparezcan. En algunos casos puede ser necesario borrar la palabra `original` en la primera línea del archivo si está allí.

Jugando adecuadamente con `texmf.cnf`, donde están los caminos de búsqueda, se configura sin problemas. Para activar una línea debe quitarse el carácter de comentario `%` al principio de la línea, para desactivarla añadir el carácter `%` al principio de la línea. Cuando se activa una línea debe desactivarse la que antes hacía esa función, si la había. Por ejemplo, si se tienen los archivos de la distribución bajo `/usr/share/texmf/`, archivos locales bajo `/usr/share/local.texmf` y archivos personales bajo `~/texmf`, las líneas del `texmf.cnf` que lo harían son

```
TEXMFMAIN = /usr/share/texmf
```

para la rama principal, que viene activada por omisión. Para las ramas local y personal se añadiría (o se quitaría el comentario de la misma) una línea del tipo

```
TEXMFLOCAL = /usr/share/texmf.local
HOMETEXMF = $HOME/texmf
```

que normalmente vienen comentadas. Finalmente se seleccionaría

```
TEXMF = {$HOMETEXMF,!!$TEXMFLOCAL,!!$TEXMFMAIN}
```

que las junta todas. Como se ha dicho antes, en la estructura de las ramas local y personal debe clonarse la estructura de la rama principal y como se dice en la sección siguiente debe correrse `texhash` (o `mktexlsr`) después de hacer los cambios para rehacer la base de datos de archivos. Para la rama personal puede ser necesario rehacer la base de datos como usuario.

El fichero de configuración está extensamente comentado con explicaciones de la función de cada una de las posibles líneas.

### Rehaciendo la base de datos de archivos instalados

El paso final consiste en decirle a  $\text{\LaTeX}$  que hay una serie de nuevos archivos que debe ser capaz de encontrar. En la mayor parte de los sistemas  $\text{\LaTeX}$  libres recientes se mantiene una base de datos de archivos instalados, para posibilitar una búsqueda más rápida. En estos sistemas es necesario actualizar esta base de datos cada vez que se instalan nuevos archivos, mediante los programas suministrados con este fin en la distribución.

#### $\text{teTeX}$ , $\text{fpTeX}$

Correr

```
texhash
```

#### $\text{web2c}$

En cualquier distribución *web2c* reciente `texhash` debiera funcionar. Si no es así, probar con

```
mktexlsr
```

#### $\text{MikTeX}$

En una distribución  $\text{MikTeX}$  anterior a la v2.0, hacer con los menús desplegables:

```
Start-> Programs-> MikTeX-> Maintenance-> Refresh database
```

o en una ventana DOS

```
initexmf -update-fndb
```

En una distribución  $\text{MikTeX}$  mayor o igual que la v2.0, hacer con los menús desplegables

```
Start-> Programs-> MikTeX 2-> MikTeX Options
```

y pulsar el botón *Update filename database*.

## 9.4.2. fancyhdr

Este paquete permite personalizar el diseño de página de tus documentos en L<sup>A</sup>T<sub>E</sub>X creado por Piet van Oostrum, de su guía propia [vO04] hemos obtenido la información necesaria para utilizar sus funciones.

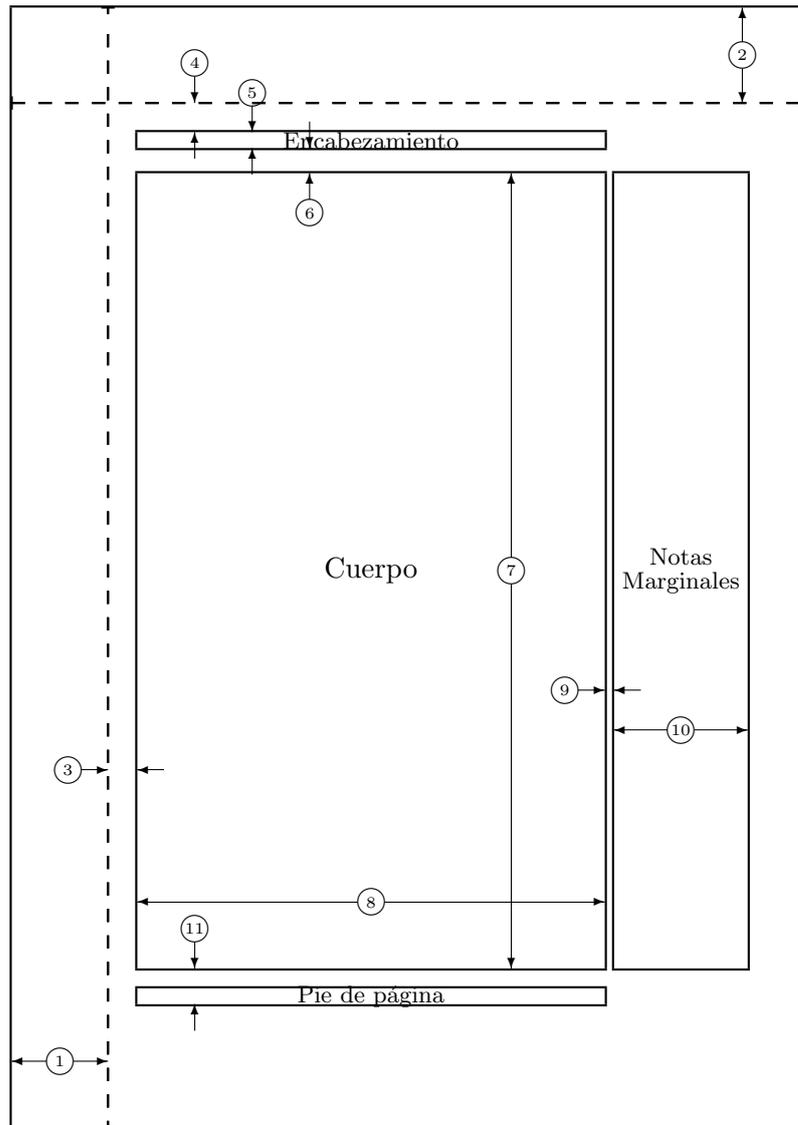
Lo podemos encontrar en el repositorio de CTAN en la siguiente dirección:

<http://ctan.org/get/macros/latex/contrib/fancyhdr.zip>

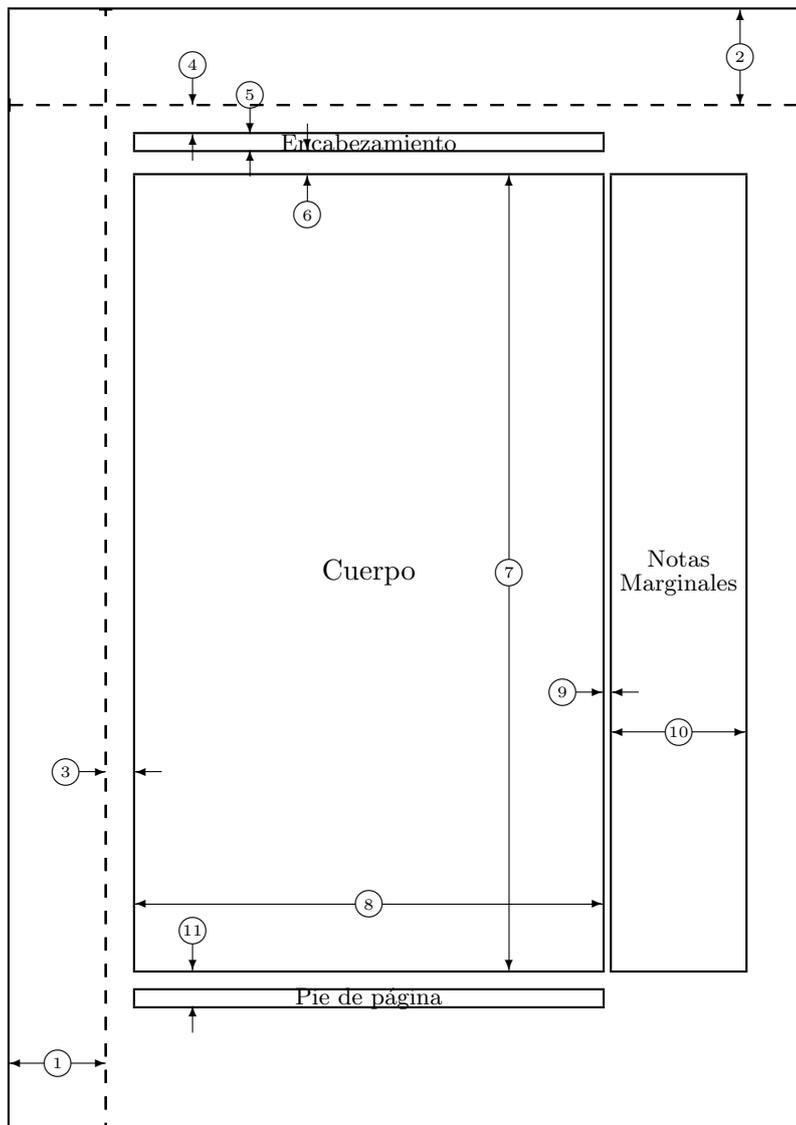
Una página de un documento en L<sup>A</sup>T<sub>E</sub>X esta constituida por varios elementos, el cuerpo que contiene el texto principal del documento junto con otros elementos llamadas flotantes (tablas y s). Las páginas son construidas por la rutina de salida de L<sup>A</sup>T<sub>E</sub>X y como esta es bastante complicada de modificar por lo tanto usaremos instrucciones para obtener los resultados deseados.

El paquete `fancyhdr` es una personalización de diseño de páginas, paquete que sustituye el paquete `fancyheadings.`, que gestiona las cabeceras y pies de página de manera eficiente, pero también trabaja con la colocación de elementos flotantes como hemos dicho. Con el paquete, puede definir encabezados y pies de página con múltiples partes y en varias líneas, especifique la cabecera y pie de página la información, normas lugar en las cabeceras y pies de página, y utilizar una cabecera y el pie ancho diferente de la del texto. Además, con el paquete puede usar distintos encabezados y pies de página, incluso para las páginas impares y, en primer lugar las páginas de los capítulos, y las páginas que contengan elementos flotantes. Puede mover la ubicación de los números de página, aumentar el espacio utilizado para las cabeceras y pies de página, y producir diccionario cabeceras de estilo que refleja la primera y la última palabra en una página. También puede suprimir los encabezados y pies de página en las páginas seleccionadas. El paquete `fancyhdr` también ofrece control sobre los tipos de letras y mayúsculas y minúsculas.

Para modificar los márgenes se utilizará `\evensidemargin` en las páginas pares y `\oddsidemargin` en las impares en el documento a doble cara, si es a una cara se utiliza `\oddsidemargin` para todo el documento, no confundir estas instrucciones con `\leftmargin` (se utiliza en la sangría de las listas).



1	una pulgada + \hoffset	2	una pulgada + \voffset
3	\oddsidemargin = 22pt	4	\topmargin = 22pt
5	\headheight = 12pt	6	\headsep = 19pt
7	\textheight = 600pt	8	\textwidth = 350pt
9	\marginparsep = 7pt	10	\marginparwidth = 100pt
11	\footskip = 27pt		\marginparpush = 5pt (no mostradas)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 597pt		\paperheight = 845pt



- |    |                        |    |                                     |
|----|------------------------|----|-------------------------------------|
| 1  | una pulgada + \hoffset | 2  | una pulgada + \voffset              |
| 3  | \oddsidemargin = 22pt  | 4  | \topmargin = 22pt                   |
| 5  | \headheight = 12pt     | 6  | \headsep = 19pt                     |
| 7  | \textheight = 600pt    | 8  | \textwidth = 350pt                  |
| 9  | \marginparsep = 7pt    | 10 | \marginparwidth = 100pt             |
| 11 | \footskip = 27pt       |    | \marginparpush = 5pt (no mostradas) |
|    | \hoffset = 0pt         |    | \voffset = 0pt                      |
|    | \paperwidth = 597pt    |    | \paperheight = 845pt                |

## Cabeceras y pies de página

Las cabeceras y pies de páginas se definen en  $\text{\LaTeX}$  con las instrucciones `\pagestyle` y `\pagenumbering` para el formato de la numeración de página.  $\text{\LaTeX}$  tiene cuatro estilos de páginas básicos:

<code>empty</code>	no hay cabecera ni pie.
<code>plain</code>	no hay cabecera, el pie tiene el número de página centrado.
<code>headings</code>	no hay pie, la cabecera contiene el nombre de capítulo/sección o subsección y número de página.
<code>myheadings</code>	no hay pie, la cabecera contiene el número de página e información del usuario.

La instrucción `\pagenumbering` define el diseño de la numeración de las páginas, y tiene los siguientes parámetros:

<code>arabic</code>	números arábigos
<code>roman</code>	números romanos en minúsculas
<code>Roman</code>	números romanos en mayúsculas
<code>alph</code>	letras en minúsculas
<code>Alph</code>	letras en mayúsculas

### ¿Qué hace `fancyhdr`?

- Cabeceras y pies en tres partes
- Líneas decorativas en cabeceras y pies
- Encabezados y pies de página más ancha que la anchura del texto
- Cabeceras y pies multilineales
- Separa cabeceras y pies para páginas pares e impares
- Diferentes cabeceras y pies para las páginas de los capítulos
- Diferentes cabeceras y pies en páginas con elementos flotantes

Para utilizar este paquete en un documento  $\text{\LaTeX} 2_{\epsilon}$ , colocamos en archivo `fancyhdr.sty` en el directorio de entrada de  $\text{\TeX}$ e incluimos en el preámbulo del documento después de `\documentclass{...}` los siguientes comandos:

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

E	Página par
O	Página impar
L	Campo izquierdo
C	Campo central
R	Campo derecho
H	Cabecera
F	Pie

Cuadro 9.3: Selectores

La configuración de los elementos de las cabeceras y pies se hace utilizando el paquete como sigue:

```
\fancyhf{} %borra todos los ajustes
\fancyhead{} %borra todos los campos de la cabecera
\fancyfoot{} %borra todos los campos de el pie
\fancyhead[R0,LE]{Cabecera1} %configuración de cabecera 1
\fancyhead[L0,RE]{Cabecera2} %configuración de cabecera 2
\fancyfoot[L0,CE]{Pie1} %configuración de pie 1
\fancyfoot[R0,CE]{Pie2} %configuración de pie 2
```

Podemos ver el diseño de página que podemos crear con fancyhdr en este ejemplo:

CabeceraIzquierda	CabeceraCentral	CabeceraDerecha
Cuerpo de Página		
PieIzquierdo	PieCentral	PieDerecho

Figura 9.1: Diseño de página fancyhdr

La Cabecera Izquierda y el Pie Izquierdo están justificados a la izquierda, la Cabecera Central el cuerpo y el Pie Central están centrados y la Cabecera Derecha y el Pie Derecho están justificados a la derecha, nosotros definimos cada uno de los seis campos y las dos líneas de separación decorativas.

### Ejemplo simple a una cara

Este es un ejemplo simple con numeración de página, título(en **negrita**), autores y nombre de proyecto.

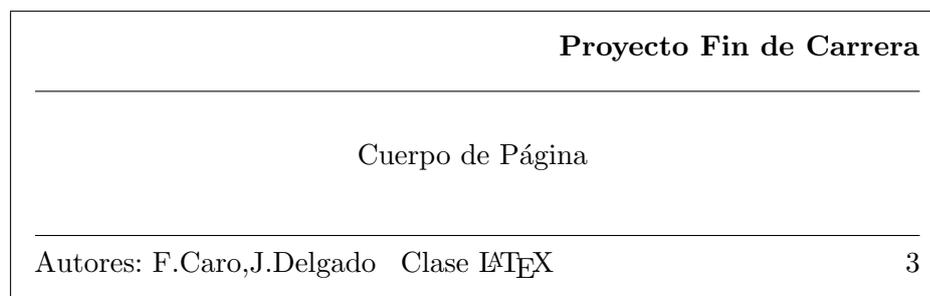


Figura 9.2: Ejemplo a una cara fancyhdr

Esto se consigue con el siguiente código:

```
\lhead{}
\chead{}
\rhead{\bfseries Proyecto fin de Carrera}
\lfoot{Autores: F.Caro,J.Delgado}
\cfoot{Clase \LaTeX{}}
\rfoot{\thepage}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}
```

La macro `\thepage` muestra el número de página actual, para eliminarlo usamos el comando `\thispagestyle{plain}` o si por el contrario no queremos cabeceras ni pie utilizaremos el comando `\thispagestyle{empty}`, y para el estilo fancy como dijimos antes utilizamos `\thispagestyle{fancy}`.

### Ejemplo simple a dos caras

Algunos tipos de documentos, tales como los basados en `book.cls` tienen por defecto la impresión a dos caras, y las páginas impares y las pares tienen diferentes diseños, en otros documentos se utiliza la opción `twoside` para imprimir a doble cara los documentos.

Ahora vamos a ver un ejemplo de diseño que se usa para páginas impares.

Esto se consigue con el siguiente código:

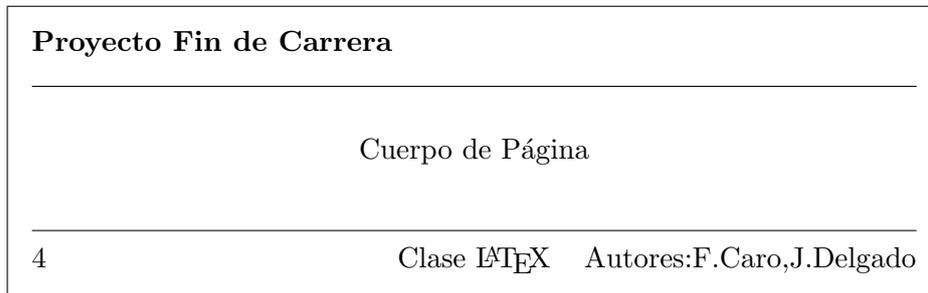


Figura 9.3: Ejemplo a dos caras fancyhdr

```

\fancyhead{} % limpia todos los campos de cabecera
\fancyhead[RO,LE]{\bfseries Proyecto Fin de Carrera}
\fancyfoot{} % limpia todos los campos de pie
\fancyfoot[LE,RO]{\thepage}
\fancyfoot[LO,CE]{Autores:F.Caro,J.Delgado}
\fancyfoot[CO,RE]{Clase \LaTeX{}}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}

```

### Diseño por defecto y modificaciones de la Clase

Para usar un documento con `book.cls` y el diseño por defecto de `fancyhdr`, basta con poner los comandos y `fancyhdr` se encarga de todo.

```

\usepackage{fancyhdr}
\pagestyle{fancy}

```

En las páginas pares, tenemos el siguiente diseño:

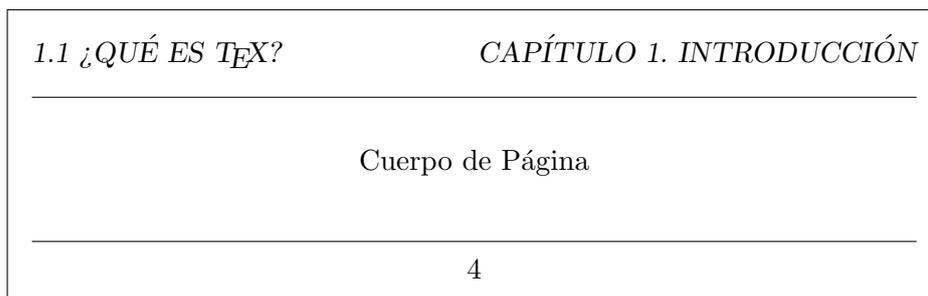


Figura 9.4: Diseño por defecto página par fancyhdr

En las páginas impares, tenemos el siguiente diseño: Este diseño por defecto lo producen las siguientes intrucciones:

```

\fancyhead[LE,RO]{\slshape \rightmark}

```

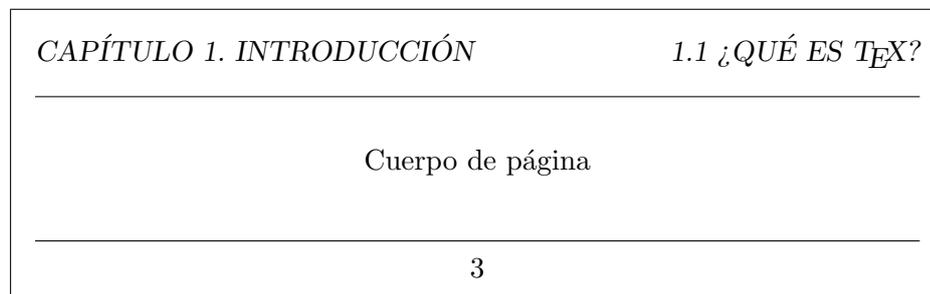


Figura 9.5: Diseño por defecto página impar fancyhdr

```
\fancyhead[LO,RE]{\slshape \leftmark}
\fancyfoot[C]{\thepage}
```

Las líneas decorativas la producen las siguientes instrucciones:

```
\headrulewidth      0.4pt
\footrulewidth      0 pt
```

Normalmente, para documentos de tipo `book` como en nuestro caso se quiere poner información de capítulos y secciones en las cabeceras de páginas,  $\text{\LaTeX}$  usa un mecanismo de marcado propio para recordar los capítulos, esto se debate a fondo en  *$\text{\LaTeX}$  Companion*, [MG94], Section 4.3.1.

Hay dos maneras posibles de cambiar la información, alto nivel o a bajo nivel, mediante las macros :

```
\leftmark (alto nivel)
\rightmark (bajo nivel)
```

Para la información de secciones a nivel bajo hemos utilizado la siguiente instrucción:

```
\renewcommand{\sectionmark}[1]{\markright{\thesection.\ #1}}
```

Esto nos cambiará “Sección 2.2 Lo que necesita saber de  $\text{\LaTeX}$ ” si es la sección actual por “2.2 Lo que necesita saber de  $\text{\LaTeX}$ ”.

Para las cabeceras hay diferentes opciones en la elección del diseño para los capítulos:

En nuestra clase hemos seleccionado el penúltimo estilo y el diseño es el siguiente

En las páginas impares, tenemos el siguiente diseño:

Code:

```

\renewcommand{\chaptermark}[1]{%
  \markboth{\chaptername
  \ \thechapter.\ #1}{}}
  Capítulo 2. Conceptos Básicos

\renewcommand{\chaptermark}[1]{%
  \markboth{\MakeUppercase{%
  \chaptername}\ \thechapter.%
  \ #1}{}}
  CAPÍTULO 2. Conceptos Básicos

\renewcommand{\chaptermark}[1]{%
  \markboth{\MakeUppercase{%
  \chaptername\ \thechapter.%
  \ #1}}{}}
  CAPÍTULO 2. CONCEPTOS BÁSI-
  COS

\renewcommand{\chaptermark}[1]{%
  \markboth{\#1}{}}
  Conceptos Básicos

\renewcommand{\chaptermark}[1]{%
  \markboth{\thechapter.\ #1}{}}
  2. Conceptos Básicos

\renewcommand{\chaptermark}[1]{%
  \markboth{\thechapter.%
  \ \chaptername.\ #1}{}}
  2. Capítulo. Conceptos Básicos

```

Prints:

Figura 9.6: Marcadores de capítulos

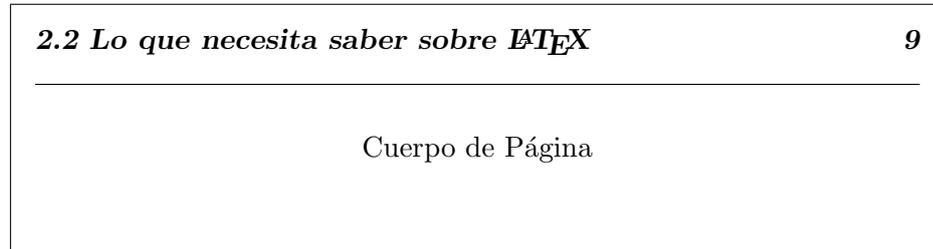


Figura 9.7: Diseño de clase para páginas impares

En las páginas pares, tenemos el siguiente diseño:

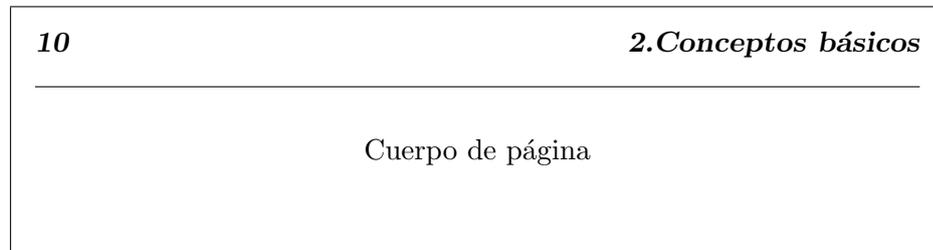


Figura 9.8: Diseño de clase para páginas pares

El código para el diseño de las páginas utilizado en nuestra clase es el siguiente:

```

1      \usepackage{fancyhdr}
2      \pagestyle{fancy}
3      \fancyhead{}
4      \fancyhead[LO]{\bfseries\slshape\nouppercase\rightmark}
5
6      \fancyhead[RE]{\bfseries\slshape\nouppercase\leftmark}
7
8      \fancyfoot{}
9      \fancyhead[LE,RO]{\bfseries\thepage}
10     \renewcommand{\headrulewidth}{0.4pt}
11     \renewcommand{\footrulewidth}{0.0pt}
12
13     \renewcommand{\chaptermark}[1]{\markboth{\thechapter.\
14     #1}{}}
14     \renewcommand{\sectionmark}[1]{\markright{\thesection.\
#1}}

```

Código 9.12: Código de Clase para diseño de cabeceras y pies de página

### 9.4.3. titlesec

Los paquetes `titlesec` and `titletoc`<sup>2</sup> han sido creados por Javier Bezos y a continuación vamos a ver como se utiliza según su guía [Bez07] y como lo hemos utilizado. Este paquete se basa en la modificación parcial o total de las macros de  $\text{\LaTeX}$  relacionadas con las secciones los nombres de títulos cabeceras y contenidos, su objetivo es proporcionar nuevas funciones inaccesibles en  $\text{\LaTeX}$  actual. Si se quiere una interfaz más vistosa que la estándar proporcionada por  $\text{\LaTeX}$  pero sin cambiar la forma de trabajo de  $\text{\LaTeX}$  se puede también utilizar el paquete `fancyhdr` mencionado anteriormente 9.4.2 de Piet van Oostrum, en `secsty` de Rowland McDonnell, o `tocloft` de Peter Wilson

Como siempre, el paquete se carga de la forma usual `\usepackage`, luego se redefinen los comando que se van a utilizar. La manera más fácil de hacer el cambio de formato es con un conjunto de opciones del paquete y un par de instrucciones. Si las opciones que nos proporciona ese conjunto de herramientas nos dan un resultado apropiado, no hace falta meterse en profundidad con instrucciones avanzadas.

Los siguientes ejemplos son ilustrativos sobre como se pueden modificar las secciones y los títulos con este paquete, estos son ejemplos de secciones y sus respectivos códigos:

■

**9.4** Este es un ejemplo de las instrucciones para la sección definido aquí abajo, otra vez más, Este es un ejemplo de las instrucciones para la sección definido aquí abajo

```
\titleformat{\section}[block]
  {\normalfont\bfseries\filcenter}
  {\fbox{\itshape\thesection}}{1em}{}
```

■

SECTION 9.5

**Un título enmarcado**

```
\titleformat{\section}[frame]
  {\normalfont}
```

<sup>2</sup>El paquete `titlesec` esta actualmente en la versión 2.8. © 1998–2007 Javier Bezos. El paquete `titletoc` esta actualmente en la versión 1.6. © 1999–2007 Javier Bezos. All Rights Reserved.

```

{\filright
 \footnotesize
 \enspace SECTION \thesection\enspace}
{8pt}
{\Large\bfseries\filcenter}

```

■

---

### 9.6. Un título con raya

```

\titelformat{\section}
{\titlerule
 \vspace{.8ex}%
 \normalfont\itshape}
{\thesection.}{.5em}{\}

```

■

---

### 9.7

#### Otro título con raya

```

\titelformat{\section}[block]
{\normalfont\sffamily}
{\thesection}{.5em}{\titlerule\ [.8ex]\bfseries}

```

■

• • •

9.8 La longitud de la “regla” por encima que es la más larga en este título esta incrementada en dos picas

• • •

9.9 Esta es una más corta

```

\titelformat{\section}[block]
{\filcenter\large
 \addtolength{\titlewidth}{2pc}%
 \titleline*[c]{\titlerule* [.6pc]{\tiny\textbullet}}}%
 \addvspace{6pt}%
 \normalfont\sffamily}
{\thesection}{1em}{\}
\titlespacing{\section}
{5pc}{*2}{*2}[5pc]

```

■

SECTION 9▷4

Este es un ejemplo de las instrucciones para la sección definido aquí abajo, otra vez más Este es un ejemplo de las instrucciones para la sección definido aquí abajo. Repitámoslo, Este es un ejemplo de las instrucciones para la sección definido aquí abajo, otra vez más, Este es un ejemplo de las instrucciones para la sección definido aquí abajo

```
\titleformat{\section}[display]
  {\normalfont\fillast}
  {\scshape section \oldstylenums{\thesection}}
  {1ex minus .1ex}
  {\small}
\titlespacing{\section}
  {3pc}{*3}{*2}[3pc]
```

■

ESTA PARTE ES EL TÍTULO EN SÍ MISMO y esto es parte de la sección...

```
\titleformat{\section}[runin]
  {\normalfont\scshape}
  {}{0pt}{}
\titlespacing{\section}
  {\parindent}{*2}{\wordsep}
```

■

**9.5. Un simple ejemplo de texto “insertado” en la forma de sección** El cual se muestra el resultado seguido de un texto. El cual se muestra el resultado seguido de un texto.El cual se muestra el resultado seguido de un texto.

**9.6. Y otro más** Observamos como el texto insertado en el título y el espacio reservado se ajusta automáticamente.Observamos como el texto insertado en el título y el espacio reservado se ajusta automáticamente.Observamos como el texto insertado en el título y el espacio reservado se ajusta automáticamente

```
\titleformat{\section}[wrap]
  {\normalfont\fontseries{b}\selectfont\filright}
  {\thesection.}{.5em}{}
\titlespacing{\section}
  {12pc}{1.5ex plus .1ex minus .2ex}{1pc}
```

■

§ 9.7. **Título runin antiguo.**—Por supuesto, si prefieres solo unos puntos suspensivos después del título, en ese caso el argumento opcional debería ser `—[.]—` y el espacio con un valor como por ejemplo, `1em`.

```
\titleformat{\section}[runin]
  {\normalfont\bfseries}
  {\S\ \thesection.}{.5em}{.---]
\titlespacing{\section}
  {\parindent}{1.5ex plus .1ex minus .2ex}{0pt}
```

■

**Ejemplo  
de  
sección al  
margen**

El cual está seguido de un texto de ejemplo para mostrar el resultado. Pero no pare de leer, porque el siguiente ejemplo muestra como aventaja a otros paquetes. La última instrucción en el último argumento puede recibir un argumento, el cual es el título sin ninguna instrucción adicional en el interior. Aquí se da el código.

```
\newcommand{\secformat}[1]{\MakeLowercase{\so{#1}}}  
  % \so spaces out letters  
\titleformat{\section}[block]  
  {\normalfont\scshape\filcenter}  
  {\thesection}  
  {1em}  
  {\secformat}
```

El margen por encima de título se define:

```
\titleformat{\section}[leftmargin]  
  {\normalfont  
  \titlerule* [.6em]{\bfseries.}%  
  \vspace{6pt}%  
  \sffamily\bfseries\filleft}  
  {\thesection}{.5em}{}  
\titlespacing{\section}  
  {4pc}{1.5ex plus .1ex minus .2ex}{1pc}
```

■

Los siguientes ejemplos están divididos por capítulos. Sin embargo este documento carece de capítulo y nos muestra como usar las secciones con un ligero cambio.

---



---

## CHAPTER 9.4

---

# El Título

```

\titleformat{\chapter}[display]
  {\normalfont\Large\filcenter\sffamily}
  {\titlerule[1pt]%
   \vspace{1pt}%
   \titlerule
   \vspace{1pc}}%
\LARGE\MakeUppercase{\chaptertitlename} \thechapter}
{1pc}
{\titlerule
 \vspace{1pc}}%
\Huge}

```

■

En nuestra clase este diseño ha sido el tipo elegido .

## CHAPTER 9

---

### El Título

---

```

\renewcommand{\thechapter}{\arabic{chapter}}
\titleformat{\chapter}[display]
  {\bfseries\Large}
  {\filleft\MakeUppercase{\chaptertitlename} \Huge\thechapter}
  {4ex}
  {\titlerule
   \vspace{2ex}%
   \filright}
  [\vspace{2ex}%
   \titlerule]

```

#### 9.4.4. tocbibind

El paquete `tocbibind` se puede usar para añadir elementos como la bibliografía o un índice a el Cuadro de Contenidos. El paquete está diseñado para trabajar con las cuatro clases estándares `book`, `report`, `article`, y algo con la clase `ltxdoc`, pero con otras clases puede resultar problemático. El paquete ha sido probado con el paquete `tocloft` según la guía (ponerla en la biblio), pero no ha sido probado con otros paquetes que cambian las definiciones de las instrucciones `\chapter*` o `\section*`. El paquete `tocbibind` provee una solución para la inserción automática de referencias a una bibliografía o a un índice, o cualquier otro elemento de cabecera de un documento en el Cuadro de Contenidos. Porciones del paquete se desarrollaron como parte de una clase y paquete para la composición de las normas tipográficas ISO.

#### Como se usa el paquete tocbibind

El paquete `tocbibind` activa los títulos de el Cuadro de Contenidos, la Lista de Figuras, la Lista de Cuadros, la Bibliografía, y el Índice de todo lo que se quiera añadir a el Cuadro de Contenidos. Por defecto todos esos elementos si existen serán incorporados automáticamente a al Tabla de Contenidos (ToC), las opciones posibles del paquete para desactivar cualquiera de estas inclusiones son:

- `notbib` Desactiva la inclusión de la Bibliografía.
- `notindex` Desactiva la inclusión del Índice.
- `nottoc` Desactiva la inclusión del Tabla de Contenidos.
- `notlot` Desactiva la inclusión de la Lista de Cuadros.
- `notlof` Desactiva la inclusión de la Lista de Figuras.
- `chapter` Utiliza en nivel de Capítulo de cabecera, si es posible.
- `section` Utiliza en nivel de Sección de cabecera, si es posible.
- `numbib` Numera la cabecera de la Bibliografía (no numerada por defecto).
- `numindex` Numera la cabecera del Índice (no numerada por defecto).
- `other` El uso no tradicional de comandos de cabecera. Esta opción eficaz requiere el uso de `\tocotherhead`.
- `none` Desactiva todo.

El paquete está diseñado para trabajar con las clases estándar de L<sup>A</sup>T<sub>E</sub>X como hemos dicho, `book`, `report`, `article`, `proc`, y `ltxdoc` la cual es una versión extendida de `article`. En las clases `article`, `proc` y `ltxdoc` L<sup>A</sup>T<sub>E</sub>X usa la instrucción `\section*` para el estilo de cabecera de la Bibliografía etc., mientras que las otras clases utilizan la instrucción `\chapter*` para el estilo de cabecera, `tocbibind` respeta esta convención, sin embargo si el paquete es utilizado con otra clase entonces las opciones de `chapter` y `section` se pueden usar para seleccionar el estilo apropiado.

Las clases estándar, con la excepción de `ltxdoc`, tienen una característica y es que la altura de el título de un índice se encuentra en una altura diferente que cualquier otro en un documento (fallo L<sup>A</sup>T<sub>E</sub>X3126). El paquete `tocbibind` desactiva esta característica. Este defecto tiene el efecto secundario de que la longitud de `\columnseprule`, `\columnsep` se puede modificar mediante el comando `\setlength` o modificar la columna de separación y el espesor de una norma entre las dos columnas en el índice. El efecto de usar la opción `none` es limitar cualquier cambio en la simple desactivación de esta característica estándar.

En las clases estándar de L<sup>A</sup>T<sub>E</sub>X la bibliografía y las cabeceras de índices están ambas definidas por la instrucción `\chapter*` o `\section*`. Este paquete asume que cualquier otra clase, otra cualquiera que no sea las estándares antes ya mencionadas, o bien utilice el código de las clases estándar para la aplicación de la bibliografía y otros títulos, o utilizará el código muy similar. Algunas clases cambian el nombre de las instrucciones de cabecera, esto sería cambiar `\section` por cualquier otra variable, y en las cabeceras del documento ocurre esto se puede usar la opción `other` y la instrucción `\tocotherhead{(headingname)}` en el preámbulo después de cargar el paquete, y este asume que la cabecera está definida por `(headingname)`.

Este paquete utiliza `tocbibname` para almacenar el nombre de la bibliografía y las siguientes instrucciones:

```
\setindexname,\settocname,\setlotname,
\setlofname,\settocbifname
```

. Cuando se usa con las clases estándares los textos de cabeceras se toman de las instrucciones

```
\indexname{ },\contentsname{ },
\listtablename{ },\listfigurename{ }
```

respectivamente. El texto de cabecera se puede cambiar cambiando instrucciones estándar o, usando `\setindexname{<name>}`, y similar para las otras cabeceras. Además, las siguientes dos líneas de código tienen el mismo efecto:

```
\renewcommand{\listfigurename}{Figures}
\setlofname{Figures}
```

<i>Índice general</i>	<b>V</b>
<i>Índice de cuadros</i>	<b>IX</b>
<i>Índice de figuras</i>	<b>XI</b>
<i>Índice de códigos</i>	<b>XIII</b>
<b>1 Introducción</b>	<b>1</b>

#### 9.4.5. listings

Este paquete nos sirve para crear listados de código fuente con diferentes diseños, y para casi todos los lenguajes usuales, tiene soporte tipográfico independiente, se pueden crear código muy parecidos a `verbatim`. Fue creado por, Carsten Heinz ©1996-2004 y Brooks Moses ©2006–2007. Lo podemos encontrar en la siguiente dirección:  
<http://www.ctan.org/tex-archive/macros/latex/contrib/listings/>

```
\documentclass{article}
\usepackage{listings}
\begin{document}
\lstset{language=Pascal}
% Insert Pascal examples here.
\end{document}
```

Figura 9.9: Mínimo ejemplo de listings

También lo podemos configurar de la siguiente forma mediante parámetros que definen los colores formas y tamaños. Finalmente llegamos a `\lstinputlisting`, el comando utilizado para imprimir archivos independientes. Tiene un argumento de opción y un argumento de nombre de archivo. Tenga en cuenta que posiblemente necesita especificar la ruta relativa al archivo. He aquí ahora el resultado que se imprime a continuación el código al pie de la letra, ya que ambos juntos no caben el texto anchura.

```
\lstinputlisting [lastline = 4] {listings.sty}
```

```
%% This is file 'listings.sty',
```

```
\lstset{basicstyle=\small,
  keywordstyle=\color{black}\bfseries\underbar,
  identifierstyle=,
  commentstyle=\color{white},
  stringstyle=\ttfamily,
  showstringspaces=false}
```

Figura 9.10: Ejemplo lstset de paquete listings

Hay una gran variedad de lenguajes, a continuación tenemos una tabla con todos los posibles lenguajes soportados por el paquete.

ABAP	ACSL	Ada	Algol
Ant	Assembler	Awk	bash
Basic	C	C++	Caml
CIL	Clean	Cobol	Comal 80
command.com	Comsol	csh	Delphi
Eiffel	Elan	erlang	Euphoria
Fortran	GCL	Gnuplot	Haskell
HTML	IDL	inform	Java
JVMIS	ksh	Lingo	Lisp
Logo	make	Mathematica	Matlab
Mercury	MetaPost	Miranda	Mizar
ML	Modula-2	MuPAD	NASTRAN
Oberon-2	OCL	Octave	Oz
Pascal	Perl	PHP	PL/I
Plasm	PostScript	POV	Prolog
Promela	PSTricks	Python	R
Reduce	Rexx	RSL	Ruby
S	SAS	Scilab	sh
SHELXL	Simula	SPARQL	SQL
tcl	TeX	VBScript	Verilog
VHDL	VRML	XML	XSLT

Cuadro 9.4: Lenguajes predefinidos para paquete listings

Este paquete no es la única utilidad para la tipografía de código fuente, hay otros paquetes que también lo hacen muy bien como:

alg, algorithm, preprin, fancyvrb.

Para nuestra clase hemos configurado el paquete de la siguiente manera, y que da como resultado lo que vemos a continuación:

```
1 \usepackage{listings}
2
3
4 \definecolor{hellgelb}{rgb}{1,1,0.8}
5 \definecolor{colKeys}{rgb}{0.6,0.15,0}
6 \definecolor{colIdentifier}{rgb}{0.7,0.1,0}
7 \definecolor{colComments}{cmyk}{0,0.3,0.99,0.25}
8 \definecolor{colString}{rgb}{0,0.5,0}
9
10 \lstset{
11     framexleftmargin=5mm,
12     frame=shadowbox,
13     rulesepcolor=\color{black},
14     float=hb,
15     basicstyle=\ttfamily\small,
16     identifierstyle=\color{colIdentifier},
17     keywordstyle=\color{colKeys},
18     stringstyle=\color{colString},
19     commentstyle=\color{colComments},
20     columns=flexible,
21     tabsize=4,
22     extendedchars=true,
23     showspaces=false,
24     showstringspaces=false,
25     numbers=left,
26     numberstyle=\tiny,
27     breaklines=true,
28     backgroundcolor=\color{hellgelb},
29     breakautoindent=true,
30     captionpos=b
31 }
```

Código 9.13: Listings

Como se puede ver en el código podemos definir cualquier color para los comentarios, las palabras reservadas, las cadenas, el color de fondo para el cuadro y el color de la sombra, la numeración de las líneas también la podemos modificar a otro tamaño de letra o quitarla, todas estas modificaciones las podemos ver mejor en la guía [?].

## 9.5— Bibliografía

Vamos a incluir una pequeña introducción sobre la economía ortotipográfica en las bibliografías por Javier Bezos y que tenemos en su nueva web.

[www.tex-tipografia.com](http://www.tex-tipografia.com).

La norma ISO <sup>3</sup> 690:1987 sobre bibliografías establece los datos que deben darse y su orden, pero deja sin definir la puntuación y el estilo de la letra (redonda, cursiva, versalitas...). Es lógico que así sea, puesto que hay muchas variaciones en función de los países y de los estilos editoriales. La norma está en inglés y por tanto emplea una puntuación acorde con la lengua en la que está escrita, pero se insiste en que no es normativa y que tiene como único fin poder dar ejemplos de forma coherente.

Las referencias se basan en normas puramente convencionales y no son texto, por lo que no se pueden aplicar directamente sus normas de puntuación, que se basan en las construcciones sintácticas o la fonética. Su función en las bibliografías es más visual que semántica, aunque el valor que tienen en el texto influye en su interpretación por el lector.

Para separar los datos, importa más la composición de cada uno de ellos que el signo de puntuación de separación, que suele pasar inadvertido; en cambio, la alternancia de redonda, versalita, cursiva y entrecomillados separa netamente unos campos de otros. También los diferentes tipos de datos, aunque no se destaquen de otra forma, permiten distinguir unos de otros: por ejemplo, el año de publicación es siempre un número.

Una vez se tienen diferenciados los campos por el estilo de la letra, no hay necesidad de introducir un nivel adicional de separación con diferentes signos de puntuación, que ya poco ayudan al lector y a cambio añaden complejidad a su preparación. Así, podríamos enunciar la idea de economía ortotipográfica: lo que ya se separa con claridad por un procedimiento no hay necesidad de separarlo por otro. Aplicado a la puntuación, el esquema más simple es separar todos los campos con un solo signo, como la coma.

Veamos algunos ejemplos:

En la 1 debe quedar claro que un autor no se identifica por una inicial, por lo que ese dato solo puede corresponder al nombre pospuesto. En la 2 se dan los nombre, pero los apellidos van en versalitas; ya tenemos la diacrisis (es decir, la distinción tipográfica) necesaria y no es necesario ir más allá. En la 3 (y la 3'), en cambio, no hay diferencia alguna entre nombres y apellidos, lo que obliga a separar más claramente los autores con punto y coma. Una variante de la primera es suprimir incluso la coma entre apellido y nombre, en el entendido de que un apellido no puede terminar

---

<sup>3</sup>La ISO (Organización Internacional de Normalización) es la entidad internacional encargada de producir los estándares normativos en los campos industriales y comerciales. La normativa ISO 690 es la que se ocupa de establecer una normativa internacional de edición de referencias bibliográficas.

con unas iniciales. La 3' ha sido muy habitual en España. (Recuérdese que la norma ISO establece que el nombre siempre sigue al apellido.)

El título siempre debería tener algún tipo de diacrisis, ya sea la cursiva, ya sea el entrecomillado, por lo que los dos puntos son redundantes. La economía en este caso se inclina por la coma. Además, los dos puntos presentan un problema: al ser de la misma altura que las minúsculas e ir pegado a la palabra precedente, se integra visualmente en ella, y en lugar de separar elementos distorsiona uno de ellos (4). La coma, al ser baja, no se integra igual y además crea un espacio adicional que ayuda a distinguir mejor la separación. El punto como separador desaparece tras una inicial (3) pero no tras nombre; en caso de nombres de una letra (que los puede haber) resulta ambiguo y por tanto es mejor evitar este signo.

El problema con los dos puntos es el mismo que en el ejemplo anterior. Dado que el orden de los datos es siempre el mismo y la editorial siempre va seguida de la fecha, no tiene por qué darse ninguna ambigüedad en el empleo de la coma, que ha sido la práctica tradicional en español (2): de nuevo la economía diacrítica se inclina por ella.

Como en todo, no hay una única manera de hacerlo. Normalmente si se trata de una bibliografía corta y no estamos interesados en reutilizar los datos bibliográficos la elección habitual es escribirla “a mano” mediante el comando `\thebibliography`. Para utilizar datos bibliográficos reutilizables, largos o complejos es preferible emplear la utilidad BibTeX.

El manejo de la bibliografía es semejante al de las referencias, de lo cual ya hemos hablado. En ambos casos, cada registro bibliográfico tiene una etiqueta. Cuando, en una determinada posición del texto, queremos hacer referencia a un registro, hacemos uso del comando `\cite{etiqueta}`. LaTeX ya se encarga de colocar el número o referencia que corresponde a esa entrada y, en el apartado de bibliografía, coloca los datos de la entrada. En caso de que queramos escribir los datos pero no queremos poner una referencia en el texto, utilizamos `\nocite{etiqueta}`.

El entorno `\thebibliography`.

- 1) PÉREZ, G., GÓMEZ, L. M., LÓPEZ, U.
- 2) PÉREZ, Gonzalo, GÓMEZ, Luis Miguel, LÓPEZ, Ulises
- 3) Pérez, Gonzalo; Gómez, Luis Miguel; López, Ulises
- 1') Pérez G., Gómez L. M., López U.
- 3') PÉREZ, GONZALO; GÓMEZ, LUIS MIGUEL; LÓPEZ, ULISES

Figura 9.11: Nombres en bibliografía

- 1) PÉREZ, G., *Título*
- 2) PÉREZ, G.: *Título*
- 3) PÉREZ, G. *Título*
- 4) PÉREZ, Gonzalo: *Título*

Figura 9.12: Nombre y título en bibliografía

- 1) New York: Dover, 2003
- 2) New York, Dover, 2003

Figura 9.13: Editorial y año en bibliografía

```
\begin{thebibliography}{n}
\bibitem{etiqueta}datos
...
\end{thebibliography}
```

donde etiqueta es la etiqueta que identifica la entrada y datos son los datos de la entrada. Por ejemplo:

```
\begin{thebibliography}{9}
\bibitem{Knuth}D. E. Knuth, The TeXbook, Addison Wesley, 1984
\end{thebibliography}
```

### BibTeX

BibTeX es un entorno más complejo para tratar bibliografía, pero es extremadamente útil y fácil de usar. Además, permite reutilizar los ficheros de bibliografía que escribamos para otros proyectos.

En primer lugar, generamos un fichero al que pondremos un nombre con extensión `.bib`; por ejemplo `mibiblio.bib`. En nuestro caso podría contener lo siguiente:

Tras esta introducción podemos crear un estilo de bibliografía propio siguiendo la norma ISO 690:1987, decir que en  $\text{\LaTeX}$  ha existido siempre el comando `\bibliography{}` con el que se especifica uno o más archivos con extensión `.bib` que contienen los datos de las referencias bibliográficas (autor, título, revista, año de edición ...) según un cierto formato. Véase, como ejemplo:

```
@BOOK{desousa,
  title = {Ortografía y ortotipografía del español actual},
  year = {2004.},
```

```

editor = {Trea},
author = {José Martínez de Sousa},
owner = {fcaro},
timestamp = {2008.08.13}
}

```

El campo @book nos indica qué tipo de registro es. Hay muchos: para libros, artículos, tesis, manuales, etc. Consulta la documentación de BibTeX para conocerlos. desousa es la etiqueta que identifica el registro que luego citaremos con \cite. El resto del archivo está claro qué es. Puede sorprender el uso de en el título. Esto se debe a que BibTeX maneja automáticamente las mayúsculas y minúsculas: el la forma de indicarle que no debe modificar lo que va dentro, pues de lo contrario lo pondrá en letras minúsculas.

Es claro que se puede editar a mano una base de datos que siga ese formato sin más que utilizar un editor de texto ASCII, pero hay mejores opciones. Para aquellos que trabajen en Windows o DOS, recomiendo el programa bibdb, LEd o WinEdt. Para Linux también hay varias posibilidades para elegir un editor decente como Emacs o Kyle, que en nuestro caso ha sido el elegido y para la creación del archivo pfcbib.bib. el cual contiene toda la información de los objetos incluidos en la bibliografía hemos utilizado **JabReF**, que es un gestor de referencias bibliográficas que permite generar bibliografías, Jabref funciona con Java, de manera que es independiente de la plataforma que se esté utilizando, es un editor gráfico de bibliografías para L<sup>A</sup>T<sub>E</sub>X, cuyo formato nativo esta basado en BiB<sub>T</sub>E<sub>X</sub> que provee una fácil interfaz para utilizar y editar los archivos de BiB<sub>T</sub>E<sub>X</sub>.

Para su ejecución en Linux JabRef requiere Java VM 1.4.2 o superior para funcionar. En el sitio web del proyecto JabRef, se puede descargar el archivo jabref-\*.jar. Una vez guardado en el directorio conveniente, se puede ejecutar normalmente desde la consola:

```
$ java -jar jabref-*.jar
```

En el punto de nuestro documento en el que queremos citar una referencia, incluimos como hemos dicho \cite{etiqueta}, donde etiqueta se refiere a la cadena de caracteres que identifica la referencia deseada dentro del archivo .bib que en el ejemplo anterior es desousa. Además en el punto en el que se quiera que aparezca, incluimos los comandos bibliographystyle{estilo.bst} y bibliography{basededatos.bib}. El primero se refiere al estilo que se desea utilizar para las referencias, tanto para la anotación que en el texto sustituye al comando \cite{etiqueta} como para la ordenación de las referencias y el formato de cada línea de la lista bibliográfica (esto es, si va el autor, luego el año, luego la revista en cursiva,...). Suelen estar disponibles en todas las instalaciones unos estilos

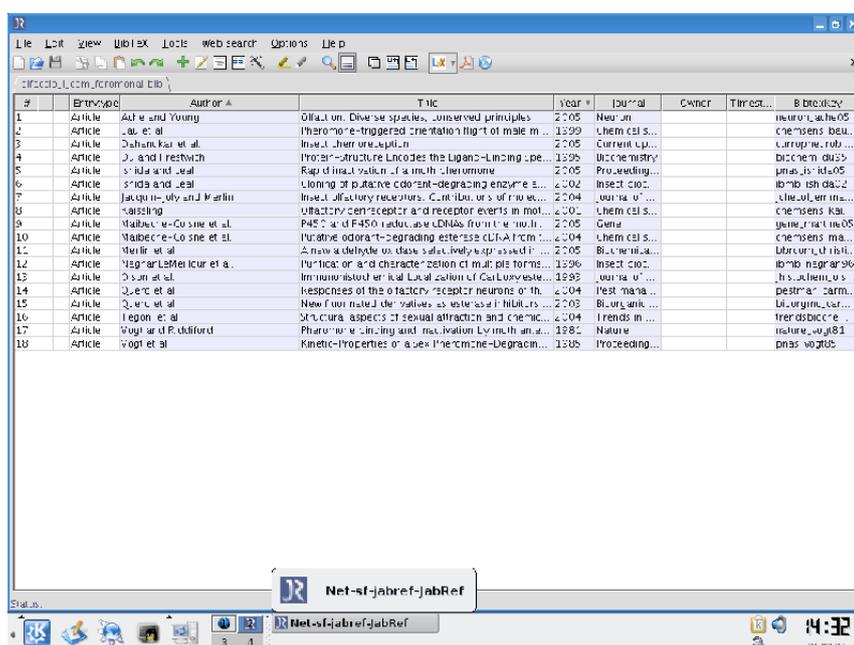


Figura 9.14: Vista general interfaz gráfica de JabRef

básicos: `plain`, `alpha`, `unsrt`, `abbrv` y hay docenas más para otras necesidades. Conviene saber dónde se guardan por defecto en nuestra instalación de modo que podamos poner otros nuevo en el mismo lugar. Éstas son las únicas adiciones que se deben hacer en el archivo fuente.

A continuación, la primera vez que se pase el archivo fuente por  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ , generará un archivo `project.aux` que recoge las citas que se hacen a lo largo del documento, así como el estilo que hemos seleccionado para la bibliografía y el nombre de la base de datos (`pfcbib.bib`).  $\text{BibT}_{\text{E}}\text{X}$  es un programa externo<sup>4</sup> que procesa el archivo `fuentes.aux` para extraer del `basededatos.bib` los datos de las citas que hayamos incluido y formatearlas según el estilo seleccionado. Se crea entonces un archivo `fuentes.bbl` que contiene un entorno `thebibliography`. La siguiente vez que procesamos el documento,  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  encuentra el archivo `fuentes.bbl` y se incluye ya la bibliografía, pero normalmente hay que procesar dos veces para que todas las referencias se incluyan correctamente. Como siempre, es interesante prestar atención a los avisos que se presentan o mirar el archivo `fuentes.log` y los que se han ido mencionando para comprobar si todo ha ido bien. Con un poco de suerte, en este punto se acaba el trabajo de incorporar la bibliografía a un documento.

<sup>4</sup>Que se ejecuta con el comando `bibtex,fuentes.aux` aunque depende de cada instalación

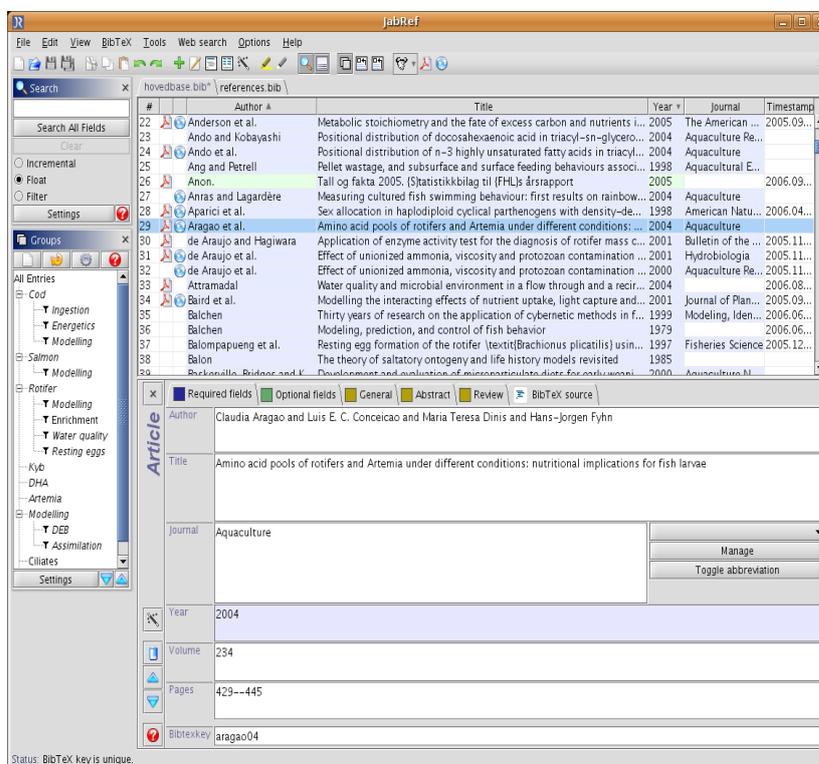


Figura 9.15: Vista de detalle interfaz gráfica de JabRef

### Custom-bib

Para dar la mayor flexibilidad en la elección de un estilo para las citas bibliográficas sin necesidad de andar cambiando la base de datos ni hacerse con infinidad de archivos de estilo, Patrick W. Daly, uno de los autores del Kopka y Daly [yPD04] ha escrito y mantiene el paquete de macros `custom-bib` que, entre otras cosas, da soporte a distintos idiomas aunque es solo válido para Linux. La última versión 4.1 de 09/08/2003 que se encuentra en la dirección <http://www.ctan.org/tex-archive/macros/latex/contrib/custom-bib>.

Un problema que enfrentan los usuarios de  $\text{BiB}\text{T}\text{E}\text{X}$  es que no existe un estándar para el formato de las listas de referencias. Editoriales y revistas insisten en una colocación de comas completamente arbitraria, dos puntos, y petición de entradas. Por otra parte, el autor (año, estilos de citas cuentan con el apoyo de algunos paquetes especiales  $\text{L}\text{A}\text{T}\text{E}\text{X}$ , pero cada uno de ellos de un número muy limitado de estilos bibliográficos. Por último, la mayoría de esos archivos de estilo son sólo para Inglés, y las adaptaciones a otros idiomas debe duplicar todo el espectro de tales archivos.

Todos estos obstáculos son, en principio, fáciles de superar simplemen-

te con una reprogramación de BiB<sub>T</sub>E<sub>X</sub> por medio de un archivo de estilo bibliográfico (extensión. BST). Bib<sub>T</sub>E<sub>X</sub> es de hecho extremadamente flexible, por desgracia, su lenguaje de programación es de muy bajo nivel, permitiendo sólo las más básicas modificaciones para el usuario normal.

La solución a esto es un nombre genérico o el archivo máster de estilo bibliográficos (extension.mbs) que contiene **docstrip** opciones alternativas para la codificación. Al seleccionar las opciones deseadas, se puede personalizar un archivo. BST archivo a su necesidades.

Este archivo, merlin.mbs, es la última versión de uno de propósito general de archivos .mbs para satisfacer la mayor cantidad de necesidades bibliográficas como sea posible. Originalmente fue por Oren Patashnik a partir de archivos estándares tales como **plain.bst** y **unsrt.bst**, además del archivo no estándar **no-apalike.bst**, un estilo muy básico autor, año citación estilo. Desde entonces, ha evolucionado ampliamente y se han añadido características que se encuentran a disposición en archivos .bst, y las exigidas por los editores o sugerido muchos usuarios / contribuyentes.

Para utilizarlo este paquete, después de poner todos los archivos que lo componen en algún directorio adecuado, y comprobar que están los archivos **makebst.ins** y varios **mbs**, luego desde la consola de Linux entramos en la carpeta correspondiente a custom-bib y ejecutamos los siguientes comandos:

1. `latex makebst.ins`

2. `latex makebst.dtx`

El primero crea el archivo **makebst.tex** y el segundo compone el archivo **makebst.dvi** y la documentación de **makebst.pdf** y **merlin.pdf**. Ahora ejecutamos el siguiente comando para que empiece la aplicación.

3. `latex makebst.tex`

El programa **docstrip**, escrito por Frank Mittelbach, es ahora una parte fundamental de la instalación en L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. Su objetivo principal es eliminar líneas de comentarios de los archivos documentados, pero tiene la función secundaria de la selección de alternativas de líneas de código. Un archivo de estilo maestro BiB<sub>T</sub>E<sub>X</sub> bibliográficos es uno que, al procesarlo por **docstrip** con las opciones seleccionadas, produce un archivo Bib<sub>T</sub>E<sub>X</sub>.BST con las características deseadas. En este sentido, **docstrip** funciona algo así como un preprocesador de C. De hecho, todos los archivos original. BST de Oren Patashnik se producen a partir de una única fuente de archivos por medio de un preprocesador. La ventaja de **docstrip** es que es portátil a todas las instalaciones con T<sub>E</sub>X.

El programa **docstrip** se puede ejecutar de forma interactiva, donde el usuario debe responder a las preguntas a través del teclado, o por medio

de un lote de puestos de trabajo donde todas las entradas se preparan en un archivo de antemano. Para la elaboración de un estilo archivo BiBTeX a partir de un archivo maestro, el método de trabajo en lote (`batch job`) es el único viable, simplemente porque el gran número de opciones disponibles. (Hay otra razón para emplear a un lote de puestos de trabajo: el método interactivo añade un comando `\endinput` al final, algo sobre lo que protesta BiBTeX.) El trabajo por lotes en general no tienen ninguna extensión, sino que utiliza `.dbj` para `docstrip`. Estos archivos son útiles porque en el documento las opciones usadas para producir el archivo `.BST`, y podrán ser modificadas para experimentar con las opciones alternativas.

1. El paquete se ejecuta y empieza a hacer preguntas. Éstas son las primeras preguntas, hay que dar el nombre del archivo de estilo que se va a crear, `pfc` en nuestro caso, la primera pregunta es si queremos un documento de la descripción del uso del archivo.

```
*****
* This is Make Bibliography Style *
*****
It makes up a docstrip batch job to produce
a customized .bst file for running with BibTeX
Do you want a description of the usage? (NO)
```

```
\yn=n
```

2. A continuación nos pregunta por un archivo MASTER de base y pondremos `merlin.mbs`.

```
Enter the name of the MASTER file (default=merlin.mbs)
```

```
\mfile=merlin.mbs
```

3. Ponemos el nombre del archivo final `.bst` que obtendremos, `pfcbibstyle` en nuestro caso.

```
Name of the final OUTPUT .bst file? (default extension=bst)
```

```
\ofile=pfcbibstyle
```

4. Damos un comentario para incluir en archivo para saber en que ámbito se aplica.

Give a comment line to include in the style file.  
Something like for which journals it is applicable.

```
\ans=Proyecto Fin de Carrera
```

5. Si quieres algunos comentarios elocuentes, ponemos que no.

Do you want verbose comments? (NO)

```
\yn=n
```

6. En esta versión de `custom-bib` tenemos el archivo `spanish.mbs` que nos proporciona la opción de crear el estilo de la bibliografía en español

```
(./merlin.mbs
<<< For more information about the meanings of
<<< the various options, see the section on
<<< Menu Information in the .mbs file documentation.
```

EXTERNAL FILES:

Name of language definition file (default=merlin.mbs)

```
\cfile=spanish.mbs
```

7. Nos pide si incluimos archivos para los nombres de diarios extras, decimos que no.

Include file(s) for extra journal names? (NO)

```
\yn=n
```

8. Seleccionamos el estilo para las citas, en nuestro caso `Alpha style` con opción de varios autores, opción (b).

STYLE OF CITATIONS:

- (\*) Numerical as in standard LaTeX
- (a) Author-year with some non-standard interface
- (b) Alpha style, Jon90 or JWB90 for single or multiple authors
- (o) Alpha style, Jon90 even for multiple authors
- (f) Alpha style, Jones90 (full name of first author)
- (c) Cite key (special for listing contents of bib file)

Select:

```
\ans=b
```

9. Para la salida de código HTML seleccionamos normal de L<sup>A</sup>T<sub>E</sub>X, opción (\*).

```
HTML OUTPUT (if non author-year citations)
(*) Normal LaTeX output
(h) Hypertext output, in HTML code, in paragraphs
(n) Hypertext list with sequence numbers
(k) Hypertext with keys for viewing databases
Select:
```

```
\ans=*
```

10. Para esta pregunta seleccionamos la opción (\*).

```
LANGUAGE FIELD
(*) No language field
(l) Add language field to switch
    hyphenation patterns temporarily
Select:
```

```
\ans=*
```

11. Seleccionamos que no queremos anotaciones, opción (\*).

```
ANNOTATIONS:
(*) No annotations will be recognized
(a) Annotations in annote field or in .tex
    file of citekey name
Select:
```

```
\ans=*
```

12. No tenemos presentaciones luego seleccionamos opción (\*).

```
PRESENTATIONS:
(*) Do not add presentation type for conference talks
(p) Add presentation, speaker not highlighted
(b) Presentation, speaker bold face
(i) Presentaion, speaker italic
(c) Presentaion, speaker in small caps
Select:
```

```
\ans=*
```

13. Seleccionamos ordenación por apellido principal, opción (x).

ORDER ON VON PART (if not citation order)  
 (\*) Sort on von part (de la Maire before Defoe)  
 (x) Sort without von part (de la Maire after Mahone)  
 Select:

\ans=x

14. Para la forma en que aparecen los nombres de los autores seleccionamos apellido, nombre completo, opción (f).

AUTHOR NAMES:  
 (\*) Full, surname last (John Frederick Smith)  
 (f) Full, surname first (Smith, John Frederick)  
 (i) Initials + surname (J. F. Smith)  
 (r) Surname + initials (Smith, J. F.)  
 (s) Surname + dotless initials (Smith J F)  
 (w) Surname + comma + spaceless initials (Smith, J.F.)  
 (x) Surname + pure initials (Smith JF)  
 (y) Surname + comma + pure initials (Smith, JF)  
 (z) Surname + spaceless initials (Smith J.F.)  
 (a) Only first name reversed, initials  
 (AGU style: Smith, J. F., H. K. Jones)  
 (b) First name reversed, with full names  
 (Smith, John Fred, Harry Kab Jones)  
 Select:

\ans=f

15. Los nombres de los editores los dejamos normal, opción (\*).

EDITOR NAMES IN COLLECTIONS (if author names reversed)  
 (\*) Editor names NOT reversed as edited by John James Smith  
 (r) Editor names reversed just like authors'  
 Select:

\ans=\*

16. Posición de Junior, opción (\*).

POSITION OF JUNIOR (if author names reversed)  
 (\*) Junior comes last as Smith, John, Jr.  
 (m) Junior between as Smith, Jr., John  
 Select:

\ans=\*

17. Elegimos el punto y coma para separar los nombre de los autores, opción (s).

PUNCTUATION BETWEEN AUTHOR NAMES:

- (\*) Author names separated by commas
- (s) Names separated by semi-colon
- (h) Names separated by slash /

Select:

\ans=s

18. Referencias adyacentes con nombres repetidos siempre presentes, opción (\*).

ADJACENT REFERENCES WITH REPEATED NAMES:

- (\*) Author/editor names always present
- (d) Repeated author/editor names replaced by dash
- (2) Repeated author/editor names replaced by 2 dashes
- (3) Repeated author/editor names replaced by 3 dashes

Select:

\ans=\*

19. Número de autores en la biliografía, los ponemos todos, opción (\*).

NUMBER OF AUTHORS IN BIBLIOGRAPHY:

- (\*) All authors included in listing
- (1) Limited authors (et al replaces missing names)

Select:

\ans=\*

20. Autores en citas, opción (\*)

AUTHORS IN CITATIONS:

- (\*) One author et al for three or more authors
- (m) Some other truncation scheme

Select:

\ans=\*

21. Tipo de fuente para las referencias de nombres de los autores, opción (\*).

TYPEFACE FOR AUTHORS IN LIST OF REFERENCES:

- (\*) Normal font for author names
- (s) Small caps authors (\sc)
- (i) Italic authors (\it or \em)
- (b) Bold authors (\bf)
- (u) User defined author font (\bibnamefont)

Select:

\ans=\*

22. Posición de la fecha al final de todo, opción (e).

DATE POSITION:

- (\*) Date at end
- (b) Date after authors
- (j) Date part of journal spec. (as 1994;45:34-40) else at end
- (e) Date at very end after any notes

Select:

\ans=e

23. Formato de la fecha precedido de coma, opción (m).

DATE FORMAT (if non author-year citations)

- (\*) Plain month and year without any brackets
- (p) Date in parentheses as (May 1993)
- (b) Date in brackets as [May 1993]
- (c) Date preceded by colon as ': May 1993'
- (d) Date preceded by period as '. May 1993'
- (m) Date preceded by comma as ', May 1993'
- (s) Date preceded by space only, as ' May 1993'

Select:

\ans=m

24. Solo pondremos en la fecha el año sin el mes, opción (x).

SUPPRESS MONTH:

- (\*) Date is month and year

(x) Date is year only  
Select:

\ans=x

25. Tipo de fuente normal para la fecha, opción (\*).

DATE FONT:  
(\* ) Date in normal font  
(b) Date in bold face  
Select:

\ans=\*

26. El título debe de tener diacrisis por lo tanto opción (i).

TITLE OF ARTICLE:  
(\* ) Title plain with no special font  
(i) Title italic (\em)  
(q) Title and punctuation in single quotes ('Title,' ..)  
(d) Title and punctuation in double quotes (''Title,') ..)  
(g) Title and punctuation in guillemets (<<Title,>> ..)  
(x) Title in single quotes ('Title', ..)  
(y) Title in double quotes (''Title'', ..)  
(z) Title in guillemets (<<Title>>, ..)  
Select:

\ans=i

27. Opción (\*).

Sigue haciendo preguntas hasta esta última: .

Finished!!  
Batch job written to file 'pfcbibstyle.dbj'  
Shall I now run this batch job? (NO)

\yn=y

Después de responder a todas estas preguntas obtendremos el archivo pfcbibstyle.bst en la carpeta custom-bib.

## 9.6— Macros

Según el texto sacado de [Val97], dado que  $\text{\LaTeX}$  es un sistema de marcado, todo documento  $\text{\LaTeX}$  contiene una serie de *marcas*, o *macros* como las llamaremos a partir de ahora, dentro del texto mismo del documento. Estas *macros* permiten especificar gran variedad de aspectos que definen la forma de presentación del documento, desde consideraciones ortográficas y tipográficas sencillas de las que hablaremos en la sección 9.8 hasta la estructura del documento. Prácticamente todas las *macros* del sistema  $\text{\LaTeX}$  se escriben con caracteres ASCII y comienzan con un barra invertida " $\backslash$ ". Hay cerca de un millar de *macros* estándar pero cada autor de  $\text{\LaTeX}$  puede definir nuevas *macros*. Por ejemplo la *macro* estándar  $\text{\textbf}{texto}$  compone su argumento *texto* en **negrita**. Uno de los usos de negrita podría ser para resaltar una palabra. Entonces el autor puede definir una nueva *macro*, **resaltar**, que produce el mismo efecto que la *macro*  $\text{\textbf}{}$ . La definición de la nueva *macro* **resaltar** se hace mediante una *macro* estándar de  $\text{\LaTeX}$ ,  $\text{\newcommand}$ :

```
\newcommand{\resaltar}[1]{\textbf{#1}}
```

donde [1] indica que la *macro*  $\text{\resaltar}{}$  tiene un argumento y  $\text{\textbf}{#1}$  indica que el argumento número 1 o primero (y en este caso único) #1 se ha de componer con negrita. Una vez hecha esta definición, se puede escribir  $\text{\resaltar}{palabra}$  dentro del texto del documento para obtener **negrita** como resultado.

La ventaja de introducir nuevas *macros* dentro de un documento  $\text{\LaTeX}$  es que éstas permiten definir nombres más comprensibles para el autor, pero también modificar de una manera fácil y rápida el efecto de una misma *macro* en todo el documento  $\text{\LaTeX}$ . Por ejemplo si se quiere hacer que la *macro* **resaltar** en vez de hacer **negrita** se haga resaltar las palabras con *cursiva* solo hay que poner:

```
\newcommand{\resaltar}[1]{\emph{#1}}
```

Es por esto por lo que es muy importante definir en el preámbulo del documento  $\text{\LaTeX}$  las *macros* para los efectos tipográficos que se han de aplicar en el texto del documento, incluso cuando éstas no son nada más que sinónimos de otras *macros* estándares de  $\text{\LaTeX}$ :

La forma general de hacer:

```
\newcommand{\macro}[n]{definición}
```

donde  $n$  es el número de argumentos o parámetros de la *macro* (9 como máximo) y *definición* puede usar los  $n$  parámetros #1, #2, #3, . . . . , #n.

La activación de un *macro* en `newcommand` da un error en  $\text{\LaTeX}$  si la *macro* que se define ya había sido definida, o si se trata de una de las *macros* predefinidas por  $\text{\LaTeX}$ . En estos casos es posible modificar la definición de la *macro* existente mediante la *macro*:

```
renewcommand
```

En general, sin embargo, no es aconsejable cambiar la definición de una *macro* predefinida de  $\text{\LaTeX}$  salvo que se conozca su funcionalidad con todo detalle y que se sepa exactamente qué uso se ha hecho en el documento  $\text{\LaTeX}$  y también que uso se quiere hacer. Resulta conveniente entonces cambiar el nombre de la *macro* por otro que todavía no haya sido utilizado.

Las *macros* de  $\text{\LaTeX}$  pueden afectar a la totalidad del texto del documento como a un párrafo o incluso palabra o carácter. Así se necesita un mecanismo para indicar el ámbito de actuación de cada *macro* introducida en el documento.

El mecanismo básico para indicar el ámbito de una *macro* es la agrupación entre llaves `{:texto}`. Por ejemplo la *macro* `\emph` sirve para dar énfasis en una parte relativamente pequeña del documento, tal como unas pocas palabras componiéndolas con letras cursivas si sus letras son redondas y componiéndolas con letras redondas si su texto es cursivo. Así `\emph palabra` da énfasis al primer carácter de *palabra* mientras que `\emph{palabra}` produce *palabra*.

Otro mecanismo para indicar el ámbito de *macros* es el uso de *macros* de inicio y fin, los llamados entornos de  $\text{\LaTeX}$ . Por ejemplo, `\begin{center} párrafo \end{center}` es un entorno que permite alinear horizontalmente el texto del párrafo incluido en el entorno.

Finalmente hay ciertas *macros* de  $\text{\LaTeX}$ , llamadas *declaraciones* que tienen un efecto global en el documento, como por ejemplo `\em`, a pesar que su efecto es local si se incluyen dentro de un grupo de llaves o de un entorno.

Toda declaración tiene asociado un entorno correspondiente, el cual lleva el mismo nombre que la declaración pero sin la barra invertida, así la declaración :

```
{\em texto}
```

es equivalente al entorno

```
\begin{em}texto\end{em}
```

El hecho de escribir entornos en lugar de declaraciones puede dar como resultado un original más fácil de leer, sobre todo cuando el texto de la declaración es bastante largo, ya que suele ser más fácil encontrar los

delimitadores de un entorno que las llaves que delimitan el ámbito de una declaración.

### 9.6.1. Macros de la clase

Con el objetivo de facilitar diversas tareas al usuario de esta clase, hemos definido en ella una serie de macros con diferentes funcionalidades. En este apartado veremos con detalle dichas macros y sus posibles utilidades.

Para comenzar tenemos definiciones sencillas para facilitar la escritura en  $\LaTeX$ . Se utilizan para facilitar a la hora de escribir palabras reservadas como  $\LaTeX$ {}, que sería lo mismo que poner `latex`. Se podría hacer lo mismo con cualquier otra palabra o frase.

```
1 \def\latex/{\protect\LaTeX{}}
2 \def\tex/{\TeX}
3 \newcommand\latexdos{\LaTeX~2.09}
```

Código 9.14: Macros sobre  $\LaTeX$

Para hacer la inserción de archivos de código fuente en múltiples lenguajes de programación según el cuadro 9.4, hemos definido la siguiente macro que tiene tres variables de entrada cuyo funcionamiento podemos ver en la imagen.

```
1 %%%%para codigo fuente%%%%%%%%
2 %1 Lenguaje segun la tabla de opciones.
3 %2 Titulo de la captura de codigo
4 %3 Ruta del archivo
5 \newcommand{\codigofuente}[3]{
6   \lstinputlisting[language=#1,caption={#2}]{#3}
7 }
```

Código 9.15: Macro para código fuente

$\LaTeX$  tiene muchas formas de incluir imágenes, para ello es necesario hacer uso del paquete:

```
\usepackage{graphicx}
```

y el comando:

```
\includegraphics[opciones]{nombre}
```

Según las necesidades se pueden insertar las imágenes en cualquier parte del documento incluso rodeado por texto, imágenes invertidas y otras muchas opciones que se pueden encontrar en cualquier manual. La siguiente macro contiene el posicionamiento y las opciones básicas de inserción de imágenes, es decir en el centro del documento, arriba o abajo según convenga, una referencia y una etiqueta. Para todas estas opciones la macro tiene cinco variables de entrada cuyo funcionamiento podemos ver a continuación.

```

1 % El comando \figura nos permite insertar figuras
2 % comodamente, y utilizando siempre el mismo formato.
3 % Los parametros son:
4 % 1 -> Porcentaje del ancho de pagina que ocupar
5 % la figura (de 0 a 1)
6 % 2 --> Fichero de la imagen
7 % 3 --> Texto a pie de imagen
8 % 4 --> Etiqueta (label) para referencias
9 % 5 --> Opciones que queremos pasarle al \includegraphics
10 \newcommand{\figura}[5]{
11   \begin{figure}
12     \begin{center}
13       \includegraphics[width=#1\textwidth,#5]{#2}%
14       \caption{#3}
15       \label{#4}
16     \end{center}
17   \end{figure}
18 }

```

Código 9.16: Macro para insertar imágenes

Los cuadros en L<sup>A</sup>T<sub>E</sub>X al igual que las figuras tiene muchísimas opciones, en nuestro caso la macro crea una tabla sencilla en la que podemos especificar el número de columnas y filas, el título de el cuadro y la etiqueta, tiene cuatro variables de entrada cuyo funcionamiento podemos ver a continuación.

```

1 %COMANDO PARA INSERTAR UN CUADRO UTILIZANDO EL FORMATO:
2 %1---> especificar numero de columnas y su alineacion ejm:
3 % |r||c|c| r=right, c=center,l=left
4 %2---> especificar el caption o titulo de la figura
5 %3---> label para hacer referencia a la tabla insertada
6 %4---> contenido de tabla separando columnas con & y filas con \\
7 \newcommand{\cuadro}[4]{
8   \begin{table}[htb]
9     \centering
10    \begin{tabular}{#1}
11      \hline
12      #4
13      \hline
14    \end{tabular}
15    \caption{#2}
16    \label{#3}
17  \end{table}
18 }

```

Código 9.17: Macro para insertar cuadros

Para la portada hemos creado macros para los datos de entrada que necesita la el comando `\hacerportada`. Hemos creado macros para autores,

titulación, título, tutor y departamento. El código es el siguiente:

```

1  %%%%%%%%% DATOS DE PORTADA %%%%%%%%%
2  \newcommand{\autores}[2]{
3      \def\@autorUno{#1}\def\@autorDos{#2}}
4
5  \newcommand{\autor}[1]{
6      \def\@autorUno{#1}}
7
8
9  \newcommand{\@autorUno}{\ClassError{pclass}{
10     Falta especificar \string\autor{}}
11     \MessageBreak Usa el comando
12     \string\autor{}} en el preambulo
13     \MessageBreak para especificar tu nombre.
14 }}
15
16
17 \newcommand{\titulacion}[1]{
18     \def\@titulacion{#1}}
19     \newcommand{\@titulacion}{\ClassError{pclass}{
20     Falta especificar \string\titulacion{}}
21     \MessageBreak Usa el comando
22     \string\titulacion{}} en el preambulo para
23     \MessageBreak especificar la titulacion que cursas.
24 }}
25
26
27 \newcommand{\titulopro}[1]{
28     \def\@titulopro{#1}}
29     \newcommand{\@titulopro}{\ClassError{pclass}{
30     Falta especificar \string\titulopro{}}
31     \MessageBreak Usa el comando
32     \string\titulopro{}} en el preambulo para
33     \MessageBreak especificar el titulo de tu proyecto.
34 }}
35
36
37 \newcommand{\tutor}[1]{
38     \def\@tutor{#1}}
39     \newcommand{\@tutor}{\ClassError{pclass}{
40     Falta especificar \string\tutor{}}
41     \MessageBreak Usa el comando
42     \string\tutor{}} en el preambulo para
43     \MessageBreak especificar el nombre del tutor de tu proyecto.
44 }}

```

Código 9.18: Macros para variables de portada

## 9.7— Entornos

Para más facilidad a la hora de utilizar el paquete `amsthm` hemos creado nuevos entornos con la traducción a español con el siguiente código:

```

1 \theoremstyle{plain}
2 \newtheorem{thm}{Teorema}[chapter]
3 \newtheorem{theorem}[thm]{Teorema}
4 \newtheorem{teorema}[thm]{Teorema}
5 \newtheorem{theoreml}[thm]{Theorem}
6
7 \newtheorem{lemma}[thm]{Lema}
8 \newtheorem{corollary}[thm]{Corolario}
9 \newtheorem{corolario}[thm]{Corolario}
10 \newtheorem{proposition}[thm]{Proposici{\ 'o}n}
11 \newtheorem{proposicion}[thm]{Proposici{\ 'o}n}
12 \newtheorem*{teoremasn}{Teorema}
13
14 \theoremstyle{definition}
15 \newtheorem{defn}{Definici{\ 'o}n}[chapter]
16 \newtheorem{definicion}[defn]{Definici{\ 'o}n}
17 \newtheorem{definition}[defn]{Definici{\ 'o}n}
18 \newtheorem{conjetura}[defn]{Conjetura}
19 \newtheorem{ejemplo}[defn]{Ejemplo}
20
21 \theoremstyle{remark}
22 \newtheorem*{remark}{Remark}
23 \newtheorem*{note}{Nota}
24 \newtheorem*{case}{Caso}
25 \newtheorem*{nota}{Nota}
26 \newtheorem*{caso}{Caso}

```

Código 9.19: Entorno para `amsthm`

También hemos creado un entorno para crear un cuadro como el utilizado en el capítulo 10.1, el código para su creación lo vemos a continuación:

```

1 \newenvironment{block}{
2   \vspace{8pt}\begin{minipage}
3     {\textwidth}}
4   {\end{minipage}\vspace{8pt}}
5 \newenvironment{fblock}{
6   \vspace{8pt}\begin{boxedminipage}
7     {\textwidth}}
8   {\end{boxedminipage}\vspace{8pt}}

```

Código 9.20: Entorno para cuadro vacío

## 9.8— Ortotipografía

La **ortotipografía** es el conjunto de usos y convenciones particulares con las que se rige la escritura por medio de elementos tipográficos en cada lengua. Se ocupa de la combinación de la ortografía y la tipografía y en particular la forma en que la primera se aplica en las obras impresas.

Martínez de Sousa [dS04] define la ortotipografía como «el conjunto de reglas de estética y escritura tipográfica que se aplican a la presentación de los elementos gráficos», como las bibliografías, cuadros, poesías, índices, notas de pie de página, citas, citas bibliográficas, obras teatrales, aplicación de los distintos estilos de letra (redonda, cursiva, versalitas, así como las combinaciones de unas y otras), etc.

Para aclarar más sobre ortografía, tipografía y ortotipografía, en qué consisten, para qué sirven y en qué se diferencian, tenemos una breve introducción de Javier Bezos en la web <http://www.tex-tipografia.com/ortotipo.html> y de la que hemos extraído este texto.

La **ortografía** es el conjunto de normas que regulan la escritura de una lengua. La ortografía decide, por ejemplo, qué letras concretas han de emplearse para escribir una palabra (como *v* o *b*, *g* o *j*...), cuándo se emplean mayúsculas, el significado básico de signos como la coma, las comillas, etc. La ortografía se aplica a todo tipo de escritos, ya sean tipográficos o caligráficos.

La **tipografía** es el arte de crear y combinar tipos, es decir, letras de imprenta, para producir libros, revistas, folletos, etc., con el objetivo de facilitar su lectura y que el contenido se transmita de forma eficaz.

La **ortotipografía** (en inglés *typographical syntax*) estudia la combinación de la ortografía y la tipografía y concreta la forma en que la primera se aplica en obras impresas. Un par de ejemplos pueden ser ilustrativos:

- la ortografía establece que las siglas han de escribirse con mayúsculas, pero un tipógrafo observará que su mayor tamaño produce «manchas» en la página que pueden distraer al lector y por tanto se introduce la norma ortotipográfica de que esas mayúsculas se pueden componer como versalitas o a un tamaño menor;
- el punto cierra oraciones, pero si coincide con una llamada de nota voladita, se pueden producir «escalones» visuales que, de nuevo, pueden distraer al lector, por lo se desplazan las llamadas para que sigan al punto.

### 9.8.1. Ortotipografía de programas informáticos

También en la web de Javier Bezos hay disponible una publicación sobre ortotipografía informática que se encuentra en su Versión 0.1. 2007-03-14 la cual tenemos a continuación:

Este documento tiene como propósito proporcionar una serie de reglas generales sobre la composición de código informático destinado a técnicos, autores, editores y correctores que tienen que tratar con obras o artículos de este tipo.

#### Código

El código escrito en algún lenguaje de programación deberá distinguirse tipográficamente del texto, normalmente con letra sin remates o mecanográfica.

#### Listados de código

En los bloques de código que ocupan varias líneas, se deberá prestar atención a la posibilidad de que la medida del texto sea inferior a la longitud de las líneas del código. Es ese caso, no se dejará que el código pase simplemente a la línea siguiente (y menos sin sangrar), sino que se organizará siguiendo las reglas sintácticas del lenguaje, normalmente con sangrado adicional, y en ocasiones con el apoyo de algún signo especial (ninguno en C, Java o Pascal, pero `\` en Python, `_` en Visual Basic y `¬` AppleScript, por ejemplo).

```
if (png_info (img)->valid & PNG_INFO_pHYs) {
    img_xres (img) = round (0.0254 *
        png_get_x_pixels(png_ptr (img), png_info (img)));
    img_yres (img) = round (0.0254 *
        png_get_y_pixels(png_ptr (img), png_info (img)));
}
```

y no

```
if (png_info (img)->valid & PNG_INFO_pHYs) {
    img_xres (img) = round (0.0254 *
png_get_x_pixels_per_meter (png_ptr (img), png_info (img)));
    img_yres (img) = round (0.0254 *
png_get_y_pixels_per_meter (png_ptr (img), png_info (img)));
}
```

También se puede emplear un signo especial que indique que una línea es continuación de la anterior y que debe considerarse que en el código

es en realidad una sola línea:

```
if (png_info (img)->valid & PNG_INFO_pHYs) {
  img_xres (img) = round (0.0254 *
    ↪png_get_x_pixels(png_ptr (img), png_info (img)));
  img_yres (img) = round (0.0254 *
    ↪png_get_y_pixels(png_ptr (img), png_info (img)));
}
```

### Alineación

Cuando la letra sea mecanográfica, se ajustarán las líneas de forma que los caracteres resulten alineados verticalmente, por lo que el espacio ha de ser fijo:

```
for f in fndf.split('\n'):
  f = f.strip()
end
```

y no

```
for f in fndf.split('\n'):
  f = f.strip()
end
```

### Código en texto

Contra la norma general, los signos de puntuación pegados al código pero que no forman parte de él, deben ser de la letra principal, aunque conservando la figura y el trazo. Además, los espacios del código, incluso si forman parte de él, deben ser los que corresponden al texto (en la letra mecanográfica suelen ser fijos y más anchos, lo que resta uniformidad a los párrafos).

Entre los métodos de la clase `String` están: *count*, *find*, *index*, *join*, *split*, *strip* y otros

y no

Entre los métodos de la clase `String` están: *count*, *find*, *index*, *join*, *split*, *strip* y otros.

La orden `a.put(x, i)` asigna `x` al valor `i`-ésimo de `a`.

y no

La orden `a.put (x, i)` asigna `x` al valor `i`-ésimo de `a`.

### Puntuación

Los listados de código, es decir, el código que no está en el texto sino dispuesto aparte, no llevarán ningún signo de puntuación que no les corresponda, incluso si les pudiera corresponder por su posición en el texto (coma, punto, etc.). Si se trata de un código breve del texto, que cita una orden, la explica, etc., sí se pondrá la puntuación que corresponde al texto.

**Licencia**

Este documento se puede distribuir e imprimir libre y gratuitamente tanto en formato electrónico como impreso, pero su contenido está bajo *copyright* del autor y no se puede copiar, ni reproducir en otras obras sin autorización previa del autor, salvo en caso de cita tal y como prevé la legislación española. ©2005-2007. Javier Bezos.

# CAPÍTULO 10

---

## Errores

---

### 10.1– Errores de compilación

En  $\text{\LaTeX}$  se pueden dividir los errores en dos tipos de mensajes básicamente:

- **Avisos:** Son mensajes que indican la existencia de algún problema no fatal, de modo que  $\text{\LaTeX}$  puede seguir analizando el documento y generando la salida (aunque probablemente ésta no sea satisfactoria). Básicamente:
  - **Undefined references** (referencias no definidas):

```
LaTeX Warning: Reference 'tab:tmn-std-informacion'
on page 234 undefined on input line 4873.
```

Son frecuentes y no necesariamente dañinos. En la primera pasada,  $\text{\LaTeX}$  normalmente no es capaz de resolver todas las referencias, por lo que es necesario dar una segunda pasada (es decir, volver a ejecutar  $\text{\LaTeX}$  sobre el documento). A veces es necesaria una tercera pasada. Cuando es necesario dar una nueva pasada,  $\text{\LaTeX}$  lo dice explícitamente: Sin embar-

```
LaTeX Warning: There were undefined references.
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
```

go, otras veces se produce por haber introducido erróneamente la etiqueta en cuestión. En estos casos, después de la segunda/tercera pasada, LaTeX sigue informando del error, así que debemos subsanarlo.

- o **Overfull hbox:** se produce cuando LaTeX no sabe cómo dividir una palabra que cae al final de una línea, y que, por tanto, invade el margen derecho. Normalmente el resultado no es aceptable; la solución pasa por buscar la palabra en cuestión e indicar cómo hay que romperla. Hay casos más peliagudos, como el ejemplo que se muestra a continuación, en el que el problema se produce en el encabezado. En este caso, romper la palabra no sirve de mucho; habría que plantearse si modificar el formato de el encabezado para permitir dos líneas, si eliminar la palabra Capítulo, si obligar a LaTeX a usar minúsculas en el encabezado, o por supuesto, cambiar el nombre del capítulo a algo más corto. Nótese que el aviso indica cuánto espacio ha invadido la línea en el margen derecho (`xx.xxxxxpt too wide`); se indica en pt, es decir, en unidades de 1/72 pulgadas. A modo de referencia, cada 10pt son aproximadamente 3,5mm, por lo que el aviso del ejemplo nos indica una invasión del margen derecho de unos 14mm.

```
Overfull \hbox (39.74638pt too wide) has occurred while
\output is active
```

- o **Underfull hbox:** se produce cuando, en el proceso de justificación, la línea queda con demasiados espacios en blanco entre palabras. Normalmente no son dañinos, pues el resultado no es demasiado desagradable a la vista. La única solución sería escribir las cosas de otra manera. La gravedad de la situación viene dada por el parámetro "`badness`"; sin embargo, incluso para valores altos puede verse que el resultado es aceptable, por ejemplo:

```
Underfull \hbox (badness 10000) in paragraph
at lines 1398-1399
```

- **Errores:** Errores. Implican que la salida no será generada. Normalmente se deben a un olvido de algún '}' o ']' o confusión entre ellos. En estos casos, LaTeX normalmente acierta al indicar la causa del error, pero no suele dar muchas pistas acerca de en qué línea está el problema. Otras veces es algún error de escritura en código o falta de algún paquete, o imagen, por ejemplo:
  - **inputenc Error:** Se suele dar cuando hay algún carácter extraño en el texto y el compilador no lo reconoce.

```
./Capitulos/capitulo3.tex:1122:Package inputenc Error:  
Keyboard character used is undefined(inputenc)  
in inputencoding 'utf8'. ...xj
```

- **missing:** Este error salta cuando falta un llave o algún paréntesis para cerrar las expresiones

```
./Capitulos/capitulo3.tex:1122:Missing } inserted. ...ux|
```

- **Cannot determine ...:** Este error salta cuando hay algún problema con el tamaño de la imagen que queremos insertar

```
Capitulos/capitulo3.tex:10:Cannot determine size  
of graphic in img/portada.jpg (no BoundingBox)
```

## 10.2— Correcciones

Una vez terminada la clase se producen errores tipográficos debido a fallos de los paquetes o del propio L<sup>A</sup>T<sub>E</sub>X. En el Títulos de Contenidos si en cualquier índice se llega a una numeración de sección

con cuatro dígitos se une con el texto sin dejar espacio. También debido al paquete fancyhdr la numeración románica de los Títulos de Contenido eran en minúscula en contra de la tipografía española que es numeración románica en mayúsculas. Estos errores los hemos solucionado incluyendo un archivo `\input{tocdef}` que contiene el siguiente código:

```
1 \makeatletter
2 \renewcommand*{\l@section{\@dottedtocline{1}{0em}{2.5em}}
3 \renewcommand*{\l@subsection{\@dottedtocline{2}{1.5em}{3.2em}}
4 \renewcommand*{\l@subsubsection{\@dottedtocline{3}{4.3em}{3.2
   em}}
5 \makeatother
6
7 \renewcommand{\frontmatter}{\pagenumbering{Roman}}
```

Código 10.1: Correcciones

# CAPÍTULO 11

---

## Manual

---

### 11.1– Manual de usuario

Ésta es la documentación de la clase `pclass.cls` que produce documentos de  $\text{\LaTeX}$  con el formato oficial definido por la Universidad de Sevilla para la asignatura proyecto informático de las titulaciones: Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas. Este documento está diseñado como un curso intensivo para que, aunque no tengas conocimiento alguno de  $\text{\LaTeX}$ , puedas aprender todos los comandos esenciales y concentrarte lo antes posible en escribir la memoria de tu proyecto de fin de carrera.

### 11.2– Introducción

Este documento es una guía para que aprendas como escribir tu memoria de proyecto de fin de carrera utilizando  $\text{\LaTeX}$ . El objetivo es que, lo más pronto posible, conozcas los comandos y las herramientas básicas del sistema. Así podrás concentrarte cuanto antes en lo realmente importante, que es escribir propiamente el material de tu proyecto.

Para tratar de ganar tiempo, y entrar lo más pronto posible en las cosas que serán importantes, se omitirán muchos detalles y definiciones rigurosas sobre lo que son y no son comandos en  $\text{\LaTeX}$ . Pueden haber incluso algunas “mentiras” en lo que dice este documento, simplemente con la idea de que, por el momento, sepas sólo lo suficiente e indispensable para poder comenzar y no te preocupes por detalles adicionales. Hecha la advertencia damos inicio a esta guía.

## 11.3— ¿Cómo consigo L<sup>A</sup>T<sub>E</sub>X?

### 11.3.1. Para Unix, Linux, etc.

Hay que instalar el paquete Te<sub>T</sub>E<sub>X</sub>. Éste incluye todo lo necesario, excepto el editor para poder escribir los documentos L<sup>A</sup>T<sub>E</sub>X. Sin embargo el paquete Te<sub>T</sub>E<sub>X</sub> no ha sido mantenido en mucho tiempo. Esto ha llevado a buscar una solución, ésta se llama TexLive. Puedes encontrarlo en <http://www.tug.org/texlive/acquire.html> e incorpora soluciones a bugs y mejoras respecto a su antecesor.

Además como se ha mencionado anteriormente, tendremos que hacer uso de un editor de texto, podemos usar:

- **Emacs** (paquete emacs), que dispone de un modo de edición especial para L<sup>A</sup>T<sub>E</sub>X, realizando los comandos. Puede ser conveniente evaluar una extensión para emacs denominada AU<sub>C</sub>TeX, que indenta automáticamente, con lo cual se obtiene una mejora ostensible de la legibilidad del código, entre otras cosas.
- **Kile**, en el cual disponemos de autocompletado de comandos L<sup>A</sup>T<sub>E</sub>X, coloreado de sintaxis, Kile automáticamente marca los comandos L<sup>A</sup>T<sub>E</sub>X y resalta los paréntesis, y puede trabajar con múltiples ficheros a la vez. Además también proporciona plantillas y patrones para facilitar la creación de documentos.

Una vez realizados estos pasos ya estás listo y puedes ir directamente a la Sección 11.4.

### 11.3.2. Para Windows

La versión más popular de L<sup>A</sup>T<sub>E</sub>X para Windows se llama MiK<sub>T</sub>E<sub>X</sub> y la puedes bajar desde <http://www.miktex.org>. Desde ahí bajas un *Setup Wizard* que, una vez instalado, se conecta a internet para bajar e instalar el resto del programa. Hay varios paquetes disponibles, puede ser recomendable usar el pequeño (small) que, por lo pronto, es más que suficiente para tus necesidades básicas.

Es muy recomendable, si quieres generar y ver tu memoria en el formato PDF (*Portable Document Format*), que tengas instalado Adobe Acrobat Reader en tú ordenador. Es muy probable que ya lo tengas instalado pero, si no lo tienes, lo puedes bajar desde <http://www.adobe.com/products/acrobat>.

Además necesitarás también los programas AFPL Ghostscript y GSview para poder manipular archivos *PostScript*. Ambos programas los puedes conseguir en la página de Internet <http://www.cs.wisc.edu/~ghost/>.

Por último, también es muy recomendable que bajes el `TEXnicCenter`. Es un editor de texto especializado para `LATEX` con botones y ventanas, muy intuitivo y fácil de usar. Este programa, altamente recomendable, lo puedes bajar en la dirección <http://www.toolscenter.org/products/texniccenter/>.

Algo importante es que el último programa que instales sea `TEXnicCenter`. Ya que, al iniciarlo la primera vez, buscará donde tienes instalados `MiKTEX` y el resto de las aplicaciones para configurar todas las opciones necesarias de manera automática.

## 11.4– Mi primer documento

El objetivo de esta sección es revisar que todos los programas que necesitas están instalados y que funcionan correctamente. Lo primero que tienes que hacer es abrir tu editor de texto plano<sup>1</sup> favorito (Text Editor, Bloc de Notas o `TEXnicCenter`).

Ahora escribe el siguiente texto en tu editor y guárdalo en un archivo con el nombre `miprimer.tex` (es importante el `.tex` al final del nombre).

```
\documentclass{article}

\author{escribe aqui tu nombre}
\title{Mi Primer Documento}

\begin{document}
\maketitle

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

Cuando lo guardes fíjate bien en que directorio lo guardaste. Es una buena recomendación, incluso, crear un directorio nuevo para cada documento distinto de `LATEX` que hagas. Si usas `TEXnicCenter` es recomendable crear un proyecto para cada documento nuevo que hagas, esto genera automáticamente una carpeta nueva para cada proyecto.

### 11.4.1. Usando `TEXnicCenter`

Si estás usando `TEXnicCenter` la cosa es muy fácil a partir de aquí. En la barra de herramientas hay una lista donde puedes escoger los diferentes modos de compilar como: `LaTeX => DVI`, `LaTeX => PS` y `LaTeX => PDF`. Escoge primero el modo `DVI` y dale click al botón para *compilar* (Build).

---

<sup>1</sup>Si tu editor de texto te permite poner formato al texto (negritas, itálicas, etc.) entonces no es de texto plano y No te servirá para escribir documentos de `LATEX`.

Verás cómo en una pequeña ventana, en la parte inferior, se muestra la información que genera  $\text{\LaTeX}$  sobre el proceso de compilado.

También, si tu documento tiene algunos errores, éstos aparecerán detallados en esta misma ventana. Lo que debes hacer es leer la descripción de los errores, tratar de entender qué está pasando, intentar corregir los errores en tu documento y entonces compilar de nuevo. **¡Nunca dejes errores sin corregir!** Si dejas que los errores se vayan acumulando en esta ventana, sólo conseguirás que  $\text{\LaTeX}$  comience a hacer tonterías y será mucho más difícil entender qué está pasando después.

Si lograste compilar con éxito, ahora puedes hacer click en el botón para *visualizar* (View). Ahora, si todo salió bien, estarás viendo tu primer documento creado en  $\text{\LaTeX}$ . Puedes intentar usar los otros modos de compilación, en PS y PDF, compilar y luego visualizar los documentos que se generan para GSview y Acrobat Reader respectivamente. Puedes saltar ahora a la Sección 11.4.3.

#### 11.4.2. Usando consolas o terminales

Si *no* estás usando  $\text{\TeX}$ nicCenter entonces tendrás que teclear los comandos de  $\text{\LaTeX}$  directamente en una terminal o consola (en Unix, Linux, etc.) o una ventana de MS-DOS o Símbolo del Sistema (en Windows). Abre una consola apropiada (según sea tu caso) y cámbiate al directorio donde hayas guardado tu archivo de prueba<sup>2</sup>.

Una vez que esté guardado tu archivo tienes que *compilarlo*. Este proceso lo que hace es tomar tu archivo (`tex`) y generar un documento (`dvi`) que ya puedes visualizar en la pantalla. Para compilar utilizas el comando `latex`:

```
> latex miprimer
This is TeX, Version 3.14159 (Web2C 7.3.1)
(miprimer.tex
LaTeX2e <1999/12/01> patch level 1
...
[1] (miprimer.aux) )
Output written on miprimer.dvi (1 page, 468 bytes).
Transcript written on miprimer.log.
```

Verás como  $\text{\LaTeX}$  te saluda y escribe alguna información en la pantalla. Si hay algún error en tu documento,  $\text{\LaTeX}$  se detendrá mostrando la información del error. Debes de leerla y tratar de entenderla, luego presiona `x` para que  $\text{\LaTeX}$  termine de trabajar, trata de corregir el error y corre `latex` de nuevo.

Si logras compilar sin errores, entonces puedes ver el documento que has creado con el comando `xdvi`:

<sup>2</sup>Normalmente se utiliza el comando “`cd directorio`” para cambiar de directorio.

```
> xdvi miprimer
```

Entonces, si todo salió bien, estarás viendo en pantalla tu primer documento creado en L<sup>A</sup>T<sub>E</sub>X. También debes conocer el comando `dvips` que se escribe:

```
> dvips -o miprimer.ps miprimer.dvi
```

Ésto genera un archivo postscript (PS) que tiene muy buena calidad de impresión. Alternativamente puedes generar archivos en formato PDF para leer con Acrobat Reader. Este archivo lo consigues compilando utilizando el comando especial `pdflatex`:

```
> pdflatex miprimer
```

Por lo pronto estos son todos los comandos que necesitas para sobrevivir en el mundo de L<sup>A</sup>T<sub>E</sub>X.

### 11.4.3. Sugerencias

El formato DVI es un formato muy rápido y cómodo para trabajar, aunque no tiene muy buena calidad de impresión y no presenta adecuadamente algunas imágenes y efectos especiales que se pueden hacer con L<sup>A</sup>T<sub>E</sub>X. Por su parte los formatos PS y PDF tienen mucha mayor calidad de impresión y son más comunes como formatos para distribuir documentos.

La mayoría de los visores de DVI, por ejemplo, te permiten dejar abierto el programa con el que estás visualizando. Si haces algún cambio en tu documento y compilas de nuevo, cuando regreses a la ventana del visualizador se actualizará automáticamente para mostrar los últimos cambios. Mientras que otros programas, como Acrobat Reader, te obligarán a cerrar el documento antes de dejarte compilarlo de nuevo.

Por su parte los documentos en PS y PDF son más comunes, y casi cualquier ordenador tendrá algún programa instalado para poder visualizarlos. Además, los archivos guardados con estos formatos no pueden modificarse, lo que los hace mucho más seguros. La moraleja de esta pequeña sección es que uses DVI para editar y PS o PDF para imprimir y distribuir versiones finales.

## 11.5– Formato de la memoria de tu proyecto

Antes de seguir debes de tener en tu ordenador la clase `pclass.cls` y todos los demás archivos relacionados. En esta clase se encuentra la información sobre el reglamento oficial de la Universidad de Sevilla para la asignatura proyecto informático de las titulaciones: Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica

en Informática de Sistemas. Además también podrás encontrar muchos comandos e instrucciones especiales que te facilitarán tu tarea mientras escribes tu memoria.

Si quieres hacer la instalación de forma correcta revisa la Sección 11.11 aquí encontrarás una lista completa de los archivos que deberían venir incluidos, así como instrucciones para instalarlos en el lugar adecuado.

### 11.5.1. Datos de tu proyecto

Entre los archivos incluidos en el paquete encontrarás un `project.tex` que puedes abrir en tu editor de texto. Si estás usando `TeXnicCenter` elige, en el menú *File*, la opción *Open Project...* y abre dicho archivo. Cuando lo hayas abierto verás algunos comandos, quizá la mayoría de ellos desconocidos, pero no te preocupes demasiado por eso en este momento.

Por ahora, como suponemos que el usuario lo que quiere es empezar lo más pronto posible a escribir la memoria de su proyecto, no analizaremos de manera detallada el contenido de este archivo. De este modo pasamos directamente a lo que sí debes de saber para poder comenzar con tu memoria. Como podrás ver en el archivo, tras el comando `\begin{document}`, hay un grupo de líneas de la forma mostrada a continuación:

```
\titulopro{Escribe aqui el titulo de tu proyecto}
\tutor{Escribe aqui el nombre del tutor del proyecto}
\departamento{Nombre del departamento}
\autores{nombre1}{nombre2}
\dia{mm/aaaa}
\titulacion{Escribe aqui el nombre de tu titulacion}
```

En estas líneas encontramos los denominados *campos*, los cuales nos sirven para indicar al documento la información particular acerca de tu proyecto. Bastará con que entre cada pareja de símbolos `{ }` escribas el valor de ese campo. Por ejemplo en `\titulopro{ }` va el título de tu proyecto, en `\tutor{ }` el nombre del tutor de tu proyecto, y así sucesivamente. Todos estos datos serán utilizados para construir la portada de tu memoria.

Como podrás ver los acentos se escriben de una manera un poco rara. Para evitar problemas con la transferencia de archivos entre sistemas operativos diferentes, `LATEX` no deja utilizar acentos de manera directa. Los acentos se escriben anteponiendo a la vocal que quieras acentuar (ya sea mayúscula o minúscula) el comando `\'`. Así puedes escribir por ejemplo: `\'a`, `\'E`, `\'o`. La única excepción es la `í` que se obtiene con el comando `\'i`. Además, los comandos `\~n` y `\~N` producen “eñes” minúsculas y mayúsculas respectivamente. No dejes que te asuste mucho esta sintaxis rara para los acentos, las cosas serán mucho más simples dentro de los distintos capítulos que conformen tu memoria.

Una vez que hayas rellenado los valores adecuados para tu memoria compílala unas dos o tres veces. **¿Dos o tres veces?** Sí,  $\text{\LaTeX}$  resuelve las referencias cruzadas, las citas bibliográficas y el mismo índice del documento en varias pasadas. Normalmente eso no debe preocuparte mucho pues, mientras estás editando, no es tan importante que las referencias estén siempre correctas, conforme vayas compilando las referencias se irán actualizando poco a poco. Lo que sí debes recordar es, antes de mandar a imprimir tu documento, compilarlo varias veces para que se resuelvan correctamente todas las referencias. Si todas las referencias están correctas no debe aparecer ningún *Warning* en la salida de  $\text{\LaTeX}$ .

Si compilas el archivo y visualizas el documento DVI o generas tu archivo PS o PDF podrás ver el resultado final. De esta forma te encontrarás con que tu memoria está formada por la portada, un resumen, la página de agradecimientos, el Índice General, un capítulo de ejemplo y finalmente una página de referencias bibliográficas.

### 11.5.2. Capítulos

Al principio del mismo archivo `project.tex` encontrarás también un grupo de instrucciones como las siguientes:

```
\frontmatter
  \input{resumen.tex}
  \input{agradecimientos} %incluye el texto de agradecimientos.tex

\tableofcontents %Indice de contenidos
  \listoftables
  \listoffigures

\mainmatter %capitulos de tu memoria con numeracion arabic
  % crea un archivo .tex por cada cap'itulo que hagas
% incluye aqu'i los cap'itulos
  \input{Capitulos/capitulo1}

\backmatter
  \bibliographystyle{pfcbibstyle}
  \bibliography{pfcbib}
```

El comando `\mainmatter` marca el inicio de la sección que contendrá todos los capítulos. Así, debes tener un archivo `.tex` diferente para cada capítulo de tu memoria dentro de la carpeta `Capitulos`. Por ejemplo, si tienes tres capítulos en archivos llamados `intro.tex`, `cap1.tex` y `cap2.tex`, entonces tu sección de capítulos debería de verse como:

La instrucción `\include{Capitulos/intro.tex}` le dice a  $\text{\LaTeX}$  que dentro de la carpeta `Capitulos` se encuentra el archivo `intro.tex` que contendrá el primer capítulo de tu memoria.

```
\mainmatter % capitulos con numeracion arabic
% crea un archivo .tex por cada cap'itulo que hagas
% incluye aqu'i los cap'itulos

\input{Capitulos/intro}
\input{Capitulos/cap1}
\input{Capitulos/cap2}
```

También podemos abrir el archivo `capitulo1.tex` que se incluye a modo de ejemplo. Allí encontraras unas cuantas líneas que te mostrarán cómo debería verse el archivo correspondiente un capítulo. La primera línea siempre debe de decir:

```
\chapter{Introducci'on}\label{intro}
```

El argumento dentro del commando `\chapter{ }` indicará el título del capítulo tal como aparecerá impreso en tu memoria. El argumento dentro de `\label{ }` es un nombre corto (sin espacios) que podrás utilizar dentro del código de tu documento para hacer referencia al número del capítulo en cuestión.

A continuación de esta línea, que tiene la información del capítulo, puedes escribir ya todo el texto que quieras. El texto se escribe tal cual, los párrafos se marcan dejando una línea en blanco entre ellos. Las sangrías se agregan automáticamente, según sea necesario, al principio de cada párrafo. Todo el espaciado lo maneja  $\text{\LaTeX}$  de forma automática. No sirve de nada dejar varias líneas vacías entre párrafos o muchos espacios en blanco entre palabras para tratar de modificar los espacios que deja  $\text{\LaTeX}$ . El sistema determina automáticamente los espacios adecuados, no trates por tanto de cambiar el tamaño de estos espacios.

También podrás ver que poner los acentos en el texto dentro de los capítulos es mucho más fácil. Lo único que tienes que hacer es poner un apóstrofe ' antes de la vocal (mayúscula o minúscula) que necesites. Por ejemplo: 'a, 'e, 'i, ... Funcionan también `~n` y `~N` para las "eñes".

Prueba escribir un poco de texto y compílalo para ver los resultados. De esta forma comprobarás que es muy sencillo escribir texto en  $\text{\LaTeX}$ .

## 11.6— Notación matemática

Esta sección contiene un curso ultra rápido de como escribir fórmulas matemáticas en tus documentos. Vamos a revisar únicamente algunas construcciones sencillas y frecuentes. Al final también puedes encontrar

algunas ideas de donde buscar información para que escribir otros símbolos. En la Sección 11.9.3 hay también indicaciones adicionales para hacer algunas otras construcciones típicas en matemáticas.

### 11.6.1. Construcciones básicas

Empezamos con lo más simple. Normalmente hay fórmulas o expresiones sencillas que se insertan directamente dentro del párrafo en el que estás escribiendo. Por ejemplo: “Una función  $f$  es inyectiva si  $f(x) = f(y)$  implica que  $x = y$ ”. Esta línea de texto se produjo simplemente escribiendo:

```
Una funci'on  $f$  es inyectiva si  $f(x) = f(y)$ 
implica que  $x = y$ 
```

La primera regla de oro sobre el contenido matemático dentro de un párrafo es que **todo el contenido matemático (y sólo el contenido matemático) va entre signos \$ \$**. Aún si se trata de una sola variable  $x$  o una función  $f$  como en el ejemplo debes ponerlos entre signos \$ \$. Y por el contrario, *nunca* utilices el contenido matemático para poner una palabra o texto en itálicas, eso es incorrecto además de que no se ve bien.

Dentro del contenido matemático puedes poner exponentes y subíndices a cualquier variable o expresión con los comandos `^` y `_` respectivamente. Puedes anidar exponentes y subíndices tanto como los necesites. A continuación se presentan algunos ejemplos sencillos.

Exponentes y subíndices

```
 $x^2$        $x^{y^z}$ 
 $x_k$        $x_{y^x}$ 
 $e^{ix}$       $x^{(y+z)}$ 
```

Algunas otras expresiones típicas se obtienen usando comandos especiales como `\frac{a}{b}` para hacer quebrados ( $\frac{a}{b}$ ) y `\sqrt{x}` para la raíz cuadrada ( $\sqrt{x}$ ). Pronto te darás cuenta que hay expresiones, más o menos complicadas, que no se verían bien si las insertamos en un sólo renglón. Por ejemplo la fórmula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (11.1)$$

no se vería adecuadamente si la dejamos dentro de las mismas líneas del párrafo. Estas fórmulas se conocen como de modo *display* y se obtienen con el entorno `equation`. El ejemplo anterior se escribió con las siguientes líneas de código:

```
... rengl'on. Por ejemplo la fórmula
```

```

\begin{equation}\label{cuadratica}
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\end{equation}
no se ver'ia adecuadamente ...

```

Uno de los efectos del entorno `equation` es que, como te habrás dado cuenta, va numerando las ecuaciones. El texto dentro del comando `\label{ }` es un nombre corto (sin espacios) que te servirá para hacer referencias más adelante al número de la ecuación.

Notas de  
redacción

Observa que **no se deben dejar renglones en blanco** entre las instrucciones `\begin{equation}`, `\end{equation}` y los renglones del párrafo. Esto es porque el párrafo no se debe interrumpir por el hecho de insertar la fórmula. Incluso, como en el ejemplo anterior, cuando insertas una fórmula se debe de mantener la estructura gramatical del enunciado.

La única excepción es cuando la fórmula que insertas queda exactamente al final del párrafo. Entonces debes agregar el punto final del párrafo al terminar la ecuación y dejar un renglón en blanco para separarlo del párrafo siguiente. Esto es lo que deberías hacer si después de muchos cálculos finalmente llegas a concluir que

$$(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3. \quad (11.2)$$

El punto final del párrafo se agrega utilizando el comando `\,.` justo después de terminar la ecuación y antes del `\end{equation}`. Nota también como la primera línea de este párrafo sí lleva sangría, mientras que las líneas que siguen después de la Ecuación 11.1 no llevan sangría. En esta ocasión se utilizó el código siguiente:

```

... llegas a concluir que
\begin{equation}\label{cubo}
(x + y)^3 = x^3 + 3 x^2 y + 3 x y^2 + y^3 \,.
\end{equation}

```

El punto final del p'arrafo ...

Referencias  
cruzadas

Con la instrucción `\ref{ }` puedes hacer referencia a los números de las ecuaciones. Por ejemplo la Ecuación 11.1 (`Ecuaci'on~\ref{cuadratica}`) sirve para encontrar las soluciones de una ecuación de segundo grado, mientras que la Ecuación 11.2 (`Ecuaci'on~\ref{cubo}`) corresponde al desarrollo de un binomio al cubo. Una de las normas de  $\text{\LaTeX}$  dicta que la palabra con que se hace referencia debe iniciar en mayúscula y estar separada por un signo `~` (y no por un espacio) del comando `\ref{ }`.

Si no te interesa numerar alguna fórmula o ecuación en particular puedes usar el entorno `displaymath`, que básicamente hace lo mismo que

equation pero sin agregar números. Así

$$\sum_{i=0}^n i = \frac{n(n+1)}{2}$$

es un ejemplo, que no está numerado, y que tiene que ver con el tema siguiente. Esta fórmula la escribimos con el código:

```
... sin agregar n'umeros. As'i
\begin{displaymath}
\sum_{i=0}^n i = \frac{n(n+1)}{2}
\end{displaymath}
es un ejemplo, que ...
```

### 11.6.2. Cuadros de símbolos

Para escribir sumatorios e integrales están los comandos `\sum` e `\int` Símbolos  
que producen los símbolos respectivos. Los límites de la suma o la in- Especiales  
tegral se escriben como subíndices y exponentes del símbolo, L<sup>A</sup>T<sub>E</sub>X los  
acomodará en su lugar adecuado.

También se pueden utilizar muchos otros símbolos especiales dentro del contenido matemático. Las letras griegas:  $\pi$ ,  $\lambda$ ,  $\Omega$ ,  $\alpha$ , las obtienes llamándolas por su nombre: `\pi`, `\lambda`, `\Omega`, `\alpha`. Para las mayúsculas sólo tienes que escribir la primera letra del nombre en mayúscula, como en `\Omega`.

Otro grupo de símbolos especiales son los nombres de funciones como:  $\sin x$ ,  $\log t$ ,  $\exp n$  que obtienes del mismo modo: `\sin x`, `\log t`, `\exp n`. Cualquier función matemática conocida la puedes obtener así por su nombre. Observa que “sin” (`\sin`) es la función seno, mientras que “sin” (`\sin`) es la multiplicación de  $s$  por  $i$  por  $n$ . Algunas notaciones como  $\lim$  te permite poner también argumentos como subíndices. Por ejemplo

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1 \quad (11.3)$$

se escribe con la línea de código

```
\lim_{x \to 0} \frac{\sin x}{x} = 1
```

La siguiente cuadro muestra un pequeño repertorio de algunos símbolos y construcciones típicas: Más símbolos  
y notaciones

<code>f \colon X \to Y</code>	$f: X \rightarrow Y$
<code>i = 1, 2, \dots, n</code>	$i = 1, 2, \dots, n$
<code>(\forall x \in A)(x \leq 0)</code>	$(\forall x \in A)(x \leq 0)$
<code>A \cap B = \emptyset</code>	$A \cap B = \emptyset$
<code>A \subset (A \cup B)</code>	$A \subset (A \cup B)$

Si además necesitas añadir algunos símbolos y construcciones adicionales de los mostrados a continuación, puedes usar el paquete `symbols.sty`.

<code>\MN = \set{1, 2, \dots}</code>	$\mathbb{N} = \{1, 2, \dots\}$
<code>\ZZ, \QQ, \RR, \CC</code>	$\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$
<code>\iprod{x}{y} = x \cdot y</code>	$\langle xy \rangle = x \cdot y$
<code>\a \land (b \lor \lnot c)</code>	$a \wedge (b \vee \neg c)$
<code>\(a \lthen b) \liff (b \lif a)</code>	$(a \rightarrow b) \leftrightarrow (b \leftarrow a)$

Esta es, por supuesto, una muestra microscópica de la cantidad de signos y símbolos especiales que puedes utilizar en  $\text{\LaTeX}$ . Una guía bastante cómoda y rápida es el  $\text{\TeX}$  Cookbook de MathPro Press<sup>3</sup> que incluye una gran variedad de símbolos y construcciones comunes. Si el símbolo que buscas no está en esta guía entonces consulta The Comprehensive  $\text{\LaTeX}$  Symbol List<sup>4</sup>. Éste es prácticamente un libro completo (58 páginas y 2266 símbolos) que incluye todos los signos y notaciones que pudieras necesitar. Al final del mismo documento se discuten incluso algunos métodos para construir tus propios símbolos si es que no los llegaras a encontrar en las listas.

Si tienes un poco más de tiempo, ahora es buen momento para que juegues y experimentes construyendo diferentes tipos de fórmulas y expresiones. Es buena idea tomar un libro de matemáticas, buscar alguna fórmula más o menos complicada y tratar de escribirla en  $\text{\LaTeX}$  utilizando lo que aprendiste en esta sección. Los usuarios de  $\text{\TeX}$ nicCenter ya habrán notado que tienen a la mano una barra de herramientas con botones para insertar muchas de las fórmulas y símbolos comunes.

## 11.7– Editando en $\text{\LaTeX}$

En estas sección trataremos de dar una idea bastante general de como escribir el cuerpo principal de la memoria de tu proyecto, haciendo un énfasis particular en la estructura del documento. Una de las máximas de  $\text{\LaTeX}$  es que el autor del documento (tú en este caso) **debe preocuparse por la estructura lógica del texto** (capítulos, secciones, teoremas, demostraciones) y no por el formato particular que esta estructura implique (negritas, centrado, letra grande).

Aunque sí existen comandos para manejar cuestiones del formato, no describiremos ninguno de ellos en este manual. Ésto es con la idea de que, siguiendo la filosofía de  $\text{\LaTeX}$ , te acostumbres a pensar en organizar los documentos por su estructura y te despreocupes por completo de las cuestiones del formato.

<sup>3</sup><http://www.csd.uu.se/documentation/tex/cookbook/>

<sup>4</sup><http://www.ctan.org/tex-archive/help/Catalogue/entries/comprehensive.html>

### 11.7.1. Secciones

Por el momento ya conoces uno de los descriptores de sección más importantes que es `\chapter{ }`, pero en L<sup>A</sup>T<sub>E</sub>X tienes también otros dos descriptores que son `\section{ }` y `\subsection{ }`. Este mismo documento sirve como ejemplo de de secciones y subsecciones. El inicio de esta sección contiene, por ejemplo, el código siguiente:

```
... las cuestiones del formato.

\subsection{Secciones}\label{secciones}
```

Por el momento ya conoces ...

Observa que los comandos `\label{ }` despues de iniciar la sección nos permiten hacer referencias cruzadas. Ahora estamos por ejemplo en la Sección 11.7.1 (Sección `\ref{secciones}`) y en la Sección 11.7.2 que sigue a revisaremos algunos entornos que nos permiten construir otras estructuras especiales.

### 11.7.2. Entornos

Los entornos son comandos especiales que ofrece L<sup>A</sup>T<sub>E</sub>X para escribir Listas párrafos o bloques de texto que denotan estructuras especiales. Los entornos `enumerate` e `itemize` producen, por ejemplo, listas enumeradas y listas con viñetas respectivamente.

1. Ésta es una lista enumerada.
2. El segundo elemento de la lista va aquí.
3. Y este será el último elemento.

El código necesario para construir una lista enumerada como la del ejemplo anterior es:

```
\begin{enumerate}
\item 'Esta es una lista enumerada.
\item El segundo elemento de la lista va aquí.
\item Y este ser'a el 'ultimo elemento.
\end{enumerate}
```

Para conseguir una lista con viñetas basta con cambiar la palabra `enumerate`, en los comandos `\begin{ }` y `\end{ }` del ejemplo, por la palabra `itemize`.

Otro grupo de entornos que son de gran utilidad están definidos en el paquete `amsthm`, incluido en los archivos que acompañan a la clase `Proposiciones` y `Teoremas`.

`pclass.cls`. Estos entornos permiten enunciar lemas, teoremas, corolarios y proposiciones en tu memoria. El siguiente es un ejemplo sencillo y el código que se utilizó para generarlo.

**Proposición 11.1.** *Si  $x \in \mathbb{R}$  entonces  $x^2 \geq 0$ .*

```
\begin{proposicion}\label{positivos}
Si  $x \in \mathbb{R}$  entonces  $x^2 \geq 0$ .
\end{proposicion}
```

Si vas a citar algún teorema o resultado famoso puedes incluir entre corchetes cuadrados [ ] el nombre de dicho teorema. A continuación podrás ver otro ejemplo y su código en L<sup>A</sup>T<sub>E</sub>X:

**Teorema 11.2** (Teorema de Fermat). *La ecuación  $a^n + b^n = c^n$ , con  $n > 2$ , no tiene soluciones enteras (no triviales).*

```
\begin{teorema}[Teorema de Fermat]\label{fermat}
La ecuación  $a^n + b^n = c^n$ , con  $n > 2$ , no tiene
soluciones enteras (no triviales).
\end{teorema}
```

Referencias  
cruzadas

Observa que estos entornos van numerando también las proposiciones y teoremas. Con el mismo comando `\ref{ }` que habíamos utilizado en la sección anterior se pueden hacer referencias al Teorema 11.2 (Teorema~\ref{fermat}) y la Proposición 11.1 (Proposición~\ref{positivos}).

En la Cuadro 11.1 puedes encontrar una lista de entornos que podrás emplear en tu memoria, los cuales se han creado en `pclass.cls` basándose en `amsthm.sty`, dichos entornos los puedes utilizar igual que en los ejemplos anteriores.

### 11.7.3. Texto enfatizado

Un comando que debes conocer, pues se utiliza con bastante frecuencia, es el comando `\emph{ }`. Este comando te servirá para *enfatizar* el texto que consideres que sea importante.

```
... sirve para \emph{enfatizar} el texto
que consideres que sea importante.
```

No pienses en este comando como un comando para poner itálicas, eso sería regresar a preocuparse por el formato y además *no* es cierto. El comando `\emph{ }` emplea un tipo de letra distinto si el contexto donde se usa ya está en itálicas (por ejemplo en los enunciados de los teoremas).

---

theorem	teorema	Teorema
lemma		Lema
corollary	corolario	Corolario
proposition	proposicion	Proposición
definition	definicion	Definición
conjetura		Conjetura
ejemplo		Ejemplo
note	nota	Nota
case	caso	Caso

---

Cuadro 11.1: Entornos basados en amsthm

Algunos paquetes cambian incluso el significado de `\emph{ }` para poner el texto con alguna fuente o color especial. Algunos paquetes para hacer presentaciones con L<sup>A</sup>T<sub>E</sub>X, por ejemplo, utilizan negritas en color rojo para representar al texto enfatizado.

Uno de los usos más comunes del comando `\emph{ }` es en las definiciones matemáticas formales donde el término nuevo, introducido por la definición, se debe de enfatizar.

**Definición 11.1.** Se dice que  $A$  es un *subconjunto* de  $B$  si para todo elemento  $x \in A$  se tiene también que  $x \in B$ .

```

\begin{definition}
Decimos que  $A$  es un \emph{subconjunto} de  $B$ 
si para todo elemento  $x \in A$  se tiene tambi'en
que  $x \in B$ .
\end{definition}

```

## 11.8– Bibliografía

Quizá al principio el procedimiento general para hacer citas bibliográficas te puede parecer un poco engorroso pero verás que, de hecho, es muy eficiente y te permite olvidarte de muchos detalles cuando tengas prisa a la hora de escribir la memoria de tu proyecto.

La idea es, mas o menos, la siguiente. Debes mantener un archivo con los datos de todos los libros que vayas utilizando, puedes pensar en este archivo como en una especie de *biblioteca virtual* que tendrás dentro de tu ordenador. De esta forma cuando hagas una cita bibliográfica dentro de un documento, lo que hace el sistema es ir a buscar la referencia a tu biblioteca, extraer todos los datos del libro al que estás citando y agregar esos datos al final de tu documento en la sección de referencias bibliográficas.

Una de las ventajas que esto supone es que, por supuesto, aunque hagas muchos documentos en L<sup>A</sup>T<sub>E</sub>X no necesitas más que de una sólo biblioteca. Otra de las ventajas es que, de manera automática, siempre se agregan los datos de todos los libros a los que hagas referencia, y *sólo los libros a los que haces referencia*. Este es uno de los puntos importantes cuando elaboras la memoria de tu proyecto y uno de los principales errores que comenten los estudiantes que realizan su memoria por ejemplo en Word. ¡Usando L<sup>A</sup>T<sub>E</sub>X no tienes que preocuparte por nada pues esto se hace solito!.

Finalmente, y quizá la razón más importante, es que de nuevo tú no te preocupas por la forma en que se escriben los datos de los libros en la sección de referencias bibliográficas, L<sup>A</sup>T<sub>E</sub>X hace eso de manera automática por tí.

### 11.8.1. Archivo de biblioteca virtual

Veamos entonces como funciona el sistema. Entre los archivos que vienen de ejemplo junto con el `pclass.cls` hay un `pfcbib.bib`, este archivo será tu *biblioteca virtual* de la que ya habíamos hablado. Si abres este archivo verás una serie de bloques como el siguiente:

```
@Article{NewCam97,
  author   = {Isaac Newton and Naomi Campbell},
  title    = {A Re-formulation of Gravity with
             Respect to Really Cool Models},
  journal  = {Jornal of Funny Physics},
  pages    = {39--78},
  volume   = {35},
  year     = {1997}
}
```

La palabra `@Article` indica que la entrada se refiere a un artículo. También existen `@InProceedings` y `@Book` por mencionar algunos. La palabra que sigue después del corchete `{`, en este caso `NewCam97`, es la palabra clave con que podrás hacer referencia a esta entrada bibliográfica. Puede ser recomendable utilizar, como en el ejemplo, las tres primeras letras de los apellidos de cada autor y dos dígitos del año de publicación. Aunque, por supuesto, esto sólo es un consejo puedes usar cualquier otro sistema de claves que te parezca funcional.

Luego vienen algunos datos que debes llenar como son el autor, el título de la referencia en cuestión, el año de publicación, etc. Hay unas cuantas observaciones que tienes que tener en cuenta cuando añadas tus propias referencias.

- Los nombres de varios autores *siempre* deben ir separados con la palabra reservada `and`. No importa si estás escribiendo en español o

si son más de dos autores en la referencia, siempre separa cada pareja de autores consecutivos usando **and**. Según el estilo bibliográfico que estés utilizando los **and**'s se cambiarán por los signos adecuados en cada caso.

- Los números de páginas, como en 39--78, se deben separar por dos guiones. Esto genera un guón con el tamaño y la separación adecuada en el documento final.
- No olvides poner una coma después del valor de cada campo, **excepto** después del último campo que no lleva coma.

En el mismo archivo `pfcbib.bib` encontrarás más ejemplos de cómo escribir los datos de tus referencias bibliográficas. Cuando agregues o modifiques los datos de tus libros dentro de este archivo no olvides guardarlo.

Si todo lo anterior te parece un poco engorroso, existe una alternativa mejor que hará que puedas olvidarte de editar directamente el archivo `pfcbib.bib`. Para ello tendrás que usar *JabRef*, es una aplicación de código abierto con la que podrás administrar fácilmente las referencias bibliográficas de libros, artículos, manuales, conferencias, tesis doctorales, folletos, etc.

JabRef

No debes tener ningún problema ya que el formato de fichero que emplea JabRef es BibTeX, el estándar bibliográfico de L<sup>A</sup>T<sub>E</sub>X. De este modo editando `pfcbib.bib` mediante JabRef, te ahorrarás la tarea de escribir manualmente todo el código descrito con anterioridad. Sólo debes indicar toda la información de cada una de tus referencias bibliográficas y dicha aplicación generará el código necesario dentro de `pfcbib.bib`. JabRef te permitirá, entre otras cosas, agrupar las referencias bibliográficas por palabras clave, localizar obras en función de un patrón de búsqueda, añadir tus propios campos, etc. Puedes encontrar JabRef por ejemplo en <http://jabref.uptodown.com/>, aunque no es difícil encontrarlo en muchos otros lugares.

### 11.8.2. Citas bibliográficas

Ahora dentro los archivos de tus capítulos, como por ejemplo en `intro.tex`, puedes utilizar el comando `\cite{ }` para hacer las referencias al material bibliográfico. Quizá dentro del archivo `intro.tex` ya habrás visto unas líneas de código que dicen:

```
En el art'iculo \cite{NewCam97} se presenta una
reformulaci'on muy curiosa de la teor'ia de
la gravedad.
```

Por otra lado, al final del archivo `project.tex` puedes encontrar un par de instrucciones de la forma:

```
\bibliographystyle{pfcbibstyle}
\bibliography{pfcbib}
```

La primera de ellas indica el estilo a usar a la hora escribir la página de referencias bibliográficas, el estilo predeterminado se llama `plain` y tiene un formato estándar. Sin embargo vamos a usar un nuevo estilo bibliográfico personalizado llamado `pfcbibstyle`, este estilo ha sido generado especialmente para realizar la memoria de tu proyecto, haciendo uso de *makebst*. Podrás encontrarlo en un archivo con nombre `pfcbibstyle.bst` adjunto con la clase `pclass.cls`.

La segunda línea lo que indica es el nombre de tu biblioteca virtual, en este caso enlazará las citas bibliográficas que aparezcan en la memoria de tu proyecto con la información almacenada en `pfcbib.bib`. De este modo se podrá generar la sección de referencias bibliográficas de tu memoria con toda la información necesaria.

Una vez que hayas añadido alguna referencia nueva, para actualizar las referencias, debes correr primero  $\text{\LaTeX}$  una vez sobre el archivo de tu memoria, en nuestro caso `project.tex` (si acabas de agregar nuevas citas bibliográficas aparecerán algunos *Warnings* indicando que no se encontraron esas referencias). Luego debes ejecutar  $\text{BibTeX}$  también sobre el archivo de tu memoria, éste es el programa que se encarga propiamente de ir a la biblioteca virtual y buscar los datos de las referencias. El comando para ejecutar  $\text{BibTeX}$  desde una terminal o consola es:

```
> bibtex project
```

Hecho esto, y si no aparecen errores, debes compilar el documento de tu memoria un par de veces más usando `latex` para que se resuelvan correctamente todas las referencias que haces a los libros.

Los usuarios de  $\text{\TeX}$ nicCenter lo tienen un poco más fácil. Busca dentro del menú *Project* el comando *Properties...* Entre las opciones disponibles marca el cuadro de *Use BibTeX*. Ahora, cada vez que presiones el botón para compilar, se ejecutará también  $\text{BibTeX}$  para actualizar las referencias bibliográficas.

## 11.9— Algunos conceptos importantes

En esta última sección se tratarán varios temas que son importantes en la elaboración de tu memoria pero, sin embargo, no es necesario que los leas por completo y con mucho detalle antes de poder iniciar tu trabajo. Quizá lo más recomendable es que te concentres ya en el material de tu memoria y, cuando te haga falta información sobre alguna de estas secciones, regreses a leer lo que necesites.

Año	Importe Venta Vinos	Litros
2006	898	523
2007	764	457
2008	779	462

Cuadro 11.2: Ventas empresa vinícola

### 11.9.1. Cuadros

Hacer cuadros en  $\text{\LaTeX}$  es mas o menos sencillo. Hay un entorno para producir cuadros llamado `tabular`. Un ejemplo es la Cuadro 11.2 que muestra una cuadro, con datos ficticios, sobre ventas de una empresa de vinos. El código utilizado es el siguiente:

```
\begin{table}
  \begin{center}
    \begin{tabular}{c|cc}
      Año & Importe Venta Vinos & Litros \\
      2006 & 898 & 523 \\
      2007 & 764 & 457 \\
      2008 & 779 & 462 \\
    \end{tabular}
  \end{center}
  \caption{Ventas Empresa Vin'icola}
  \label{vinos}
\end{table}
```

Observa como los cuadros, así como las figuras que veremos en la siguiente sección, se colocan automáticamente al principio o al final de la página, como se hace en los libros reales. Si haces varios cuadros o figuras  $\text{\LaTeX}$  buscará la forma más adecuada de acomodarlas todas entre las páginas de modo que aparezcan lo más cerca posible a donde haces referencia a ellas y optimizando la calidad del resultado visual obtenido.

Las letras `{c|cc}` que aparecen después de `\begin{tabular}` indican el número y la alineación de las columnas. Debes de poner una letra por cada columna que necesites, ya sea `l`, `c`, `r` si quieres una columna alineada a la izquierda, centrada o alineada a la derecha respectivamente. En este caso usamos tres columnas centradas. El símbolo `|` separando a la primera letra de las dos siguientes indica que esas columnas deben ir separadas por una línea.

Dentro del contenido de el cuadro, se utilizan signos `&` y `\\` para separar columnas y renglones respectivamente. El comando `\hrulefill` después del final de un renglón sirve para dibujar líneas horizontales.

Como habrás observado, los cuadros y figuras se van numerando. Como siempre el comando `\label{ }` te sirve para asignar etiquetas y luego

Macro  
cuadro

`\ref{ }` para hacer las referencias.  $\text{\LaTeX}$  te ofrece además la opción de hacer índices de cuadros y figuras automáticamente. Los detalles para generar estos índices los puedes encontrar en la Sección 11.10.

También puedes usar el comando `\cuadro` para insertar cuadros en tu documento. Este comando es una macro definida en `pclass`, haciendo uso de ella insertar una cuadro te será mucho mas sencillo. Al hacer uso de este comando deberás especificar una serie de argumentos. Estos argumentos son los enumerados a continuación y siempre debes introducirlos en el mismo orden en el que aparecen.

1. **Numero de columnas y alineación**, indican el número y la alineación de las columnas. Debes de poner una letra por cada columna que necesites, ya sea `l`, `c`, `r` si quieres una columna alineada a la izquierda, centrada o alineada a la derecha respectivamente. En este caso usamos tres columnas centradas. El símbolo `|` separando a la primera letra de las dos siguientes indica que esas columnas deben ir separadas por una linea.
2. **Título de la cuadro**, esta es la caption o titulo de la cuadro.
3. **Etiqueta** (label) para hacer referencias a la cuadro insertada.
4. **Contenido de la cuadro**, separando columnas con `&` y filas con `\\`.

### 11.9.2. Figuras

Insertar figuras en  $\text{\LaTeX}$  puede ser a veces un poco engorroso, sobre todo por la gran cantidad de formatos diferentes de imágenes que existen. Por eso discutiremos aquí varias ideas para que facilitarte la vida cuando trates de insertar figuras en tu memoria.

Insertar  
Figuras

La solución es generar imágenes en formato `.eps` si utilizas postscript (PS) e imágenes en formato `.png` o `.pdf` si planeas utilizar Acrobat Reader (PDF). Una vez que tengas el archivo con la imagen que quieres insertar debes colocar el siguiente código en tu documento de  $\text{\LaTeX}$ :

```
\begin{figure}
  \begin{center}
    \includegraphics{ejemfigura}
  \end{center}
  \caption{figura de ejemplo}
  \label{ejmfigura}
\end{figure}
```

El comando `\includegraphics{ }` lleva el nombre del archivo que contiene la imagen que vas a insertar, en este caso `ejemfigura`. No necesitas poner la extensión (`.eps`, `.png` o `.pdf`) en el nombre del archivo. La

clase `pclass.cls` detecta automáticamente si estás generando archivos PS o PDF y utiliza el archivo con el formato apropiado. Lo más recomendable es generar para cada imagen dos archivos, uno en cada formato, y el sistema utilizará la versión adecuada en cada caso.

El comando `\caption{ }` indica el nombre de la figura como aparecerá impresa en el documento y `\label{ }`, como siempre, es el nombre corto para poder hacer referencias cruzadas con el comando `\ref{ }`.

Con el fin de facilitar esta tarea, el usuario puede utilizar el comando `\figura`. Esta es una macro definida en la clase `pclass` a la cual se le pasarán los argumentos enumerados a continuación y siempre en este orden: Macro figura

1. **Porcentaje del ancho de la página** que ocupará la figura, siempre será un valor entre 0 y 1.
2. **Archivo** correspondiente a la imagen que se quiere insertar.
3. **Texto para el pie** de la figura.
4. **Etiqueta** (label) para hacer referencias a la figura insertada.
5. **Opciones** que queramos pasarle al `\includegraphics`.

Ahora, ¿Cómo genero figuras en formatos `.eps`, `.png` o `.pdf`? Hay tres soluciones principales que, nos han dado buenos resultados.

La primera solución consiste en utilizar algún editor profesional de gráficos como Corel Draw, Photo Shop, Paint Shop Pro o Adobe Illustrator. Estos programas te permiten exportar imágenes en una gran variedad de formatos entre los cuales encontrarás seguramente `.eps` y `.png`. Del mismo modo, si ya tienes tu figura pero en algún otro formato incompatible (como son `jpg`, `gif`, `bmp`, `wmf`) puedes abrirla en alguno de estos editores y guardarla de nuevo en un formato adecuado. Algunos programas especializados en matemáticas como Mathematica o Matlab también pueden producir gráficos en formato `eps`. Productos Comerciales

Si no cuentas con ninguno de estos editores comerciales, y no estás interesado en comprar ninguno de ellos, existen también algunas soluciones alternativas. Una de ellas, que por su simplicidad da buenos resultados, es utilizar el editor de imágenes de StarOffice que te permite exportar en formato `.epd`. Existe también un programa llamado XnView, con diferentes versiones para Windows, Unix y muchas otras plataformas, que te permite convertir imágenes entre diferentes formatos, en particular soporta `eps` y `png`. El programa está disponible en la dirección <http://perso.wanadoo.fr/pierre.g/xnview/enxnview.html>.

Una última solución es utilizar el lenguaje nativo de L<sup>A</sup>T<sub>E</sub>X para escribir documentos. Un editor bastante estable y muy práctico es JPicEdt, disponible en <http://www.jpicedt.org>, que produce figuras usando el Lenguaje Nativo

mismo lenguaje de  $\text{\LaTeX}$ . Entre las ventajas de esta solución es que produce imágenes de muy buena calidad (sobre todo el modo `pstricks`), con el mismo tipo de letra y flexibilidad de insertar símbolos que el mismo  $\text{\LaTeX}$ . Desde luego el programa no compite con las mejores de las soluciones comerciales, pero si les da muy buena batalla. La ‘Jén el nombre del programa significa que debes de tener Java instalado en tu ordenador para usarlo.

epstopdf

Si logras crear alguna imagen en formato `.eps` es relativamente sencillo convertir la imagen a `.pdf`. El comando `epstopdf` se encarga de realizar este trabajo.

### 11.9.3. Conceptos matemáticos útiles

Aquí explicaremos algunas construcciones matemáticas que te podrían ser de utilidad. Si tienes problemas con alguno de estos ejemplos o hay alguna otra fórmula que no sabes como construir puedes consultar la Sección 11.9.4, allí encontrarás algunas pistas de lugares donde puedes obtener aún más información.

Paréntesis  
Grandes

Si necesitas poner entre paréntesis alguna expresión complicada, que ocupe más de un renglón, necesitas los comandos `\left#` y `\right#`. Estos miden el tamaño de la fórmula contenida entre ellos y la encierra con los paréntesis que tú indiques. Puedes colocar, en lugar del símbolo `#` en estos comandos, cualquiera de los siguientes: `(, )`, `\{, \}`, `[, ]`, `\langle (()`, `\rangle (())`.

```
B = \alpha \left(\frac{a + b}{d - c}\right)^2
```

$$B = \alpha \left( \frac{a + b}{d - c} \right)^2$$

Matrices

Puedes también construir matrices utilizando el entorno `array`, el siguiente ejemplo muestra cómo puedes hacerlo. Observa como los paréntesis alrededor de la matriz se colocan con los comandos `\left(` y `\right)`.

```
A = \left(
  \begin{array}{ccc}
    1 & 0 & 0 \\
    0 & 1 & 0 \\
    0 & 0 & 1
  \end{array}
\right)
```

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

En el entorno `array` las letras `{ccc}` indican el número de columnas y su alineación. Debe haber una letra por cada columna que necesites. Y cada letra puede ser ya sea `l`, `c` o `r` para indicar izquierda, centrado o derecha respectivamente. Dentro de cada renglón se separan las columnas

usando `&` y los renglones con `\\`. Observarás que los entornos `array` y `tabular` son muy similares, la diferencia es que el primero de ellos se utiliza dentro de fórmulas matemáticas, mientras que el segundo es para texto corriente.

Mezclando arreglos y paréntesis puedes conseguir otras construcciones interesantes. El siguiente es un ejemplo que podría serte de utilidad, es la construcción típica que se usa para denotar una función definida por partes. Funciones  
por Partes

```
f(x) = \left\{
  \begin{array}{ll}
    0 & x \leq 0 \\
    x^2 & 0 \leq x \leq b \\
    b^2 & \text{en otro caso}
  \end{array}
\right.
```

Observa el uso de `\text{ }` para insertar texto dentro de la fórmula y el comando `\right.` para indicar el final de la fórmula pero sin dibujar ningún paréntesis.

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x^2 & 0 \leq x \leq b \\ b^2 & \text{en otro caso} \end{cases}$$

#### 11.9.4. Alguna información de interés

Si tienes alguna duda, comentario, problema o sugerencia usando  $\text{\LaTeX}$ , algunas buenas fuentes fiables de información son:

**CervanTeX** Puedes localizarlo en <http://www.cervantex.org>. Es un grupo de usuarios hispanohablantes de  $\text{\TeX}$ ,  $\text{\LaTeX}$  y programas similares. Hay una sección de preguntas frecuentes y manuales, así como un grupo de correo donde puedes enviar tus preguntas específicas. Suscribirse al grupo de correo es más fácil que nunca, solo tienes que entrar a la página de Internet <http://filemon.mecanica.upm.es/CervanTeX/listas.php> y dejar los datos de tu correo electrónico. Te recomiendo usar el modo *digest* para no saturar demasiado tu cuenta de correo. Ya que estés suscrito puedes enviar cualquier pregunta que tengas sobre  $\text{\LaTeX}$  y la comunidad de usuarios tratará de ayudarte. Es muy probable que de un día a otro consigas varias soluciones a tu problema.

**MiKTeX** En la página de  $\text{\MiKTeX}$ , <http://www.miktex.org>, hay también referencias y manuales pero en inglés. También mantienen una

lista de usuarios donde puedes conseguir ayuda sobre cualquier problema que tengas. Para subscribirte a la lista sólo tienes que entrar a la página de Internet <http://lists.sourceforge.net/lists/listinfo/miktex-users>.

$\LaTeX$ : *A Document Preparation System, User's Guide and Reference Manual*, es un libro escrito por Leslie Lamport (el creador mismo de  $\LaTeX$ ) y publicado por Addison Wesley. Es una excelente fuente de referencia, hay muchos ejemplos y guías rápidas para solucionar problemas.

## 11.10— Información de la clase `pclass.cls`

### Campos

La siguiente es la lista de los campos disponibles en la clase `pclass`, los cuales tendrás que rellenar para configurar los datos de tu memoria. Estos campos los puedes encontrar después de la instrucción `\begin{document}` dentro de el archivo `project.tex`.

`\titulopro{Titulo}` Utiliza *Titulo* como el título que aparecerá en la memoria de tu proyecto. Puedes utilizar `\\` para indicar a  $\LaTeX$  donde iniciar una nueva línea en caso de que el título sea demasiado largo.

`\autor{Nombre}` Este campo contiene el nombre del autor de la memoria.

`\autores{Nombre1}{Nombre2}` Este campo es equivalente autor, pero lo usaremos si la memoria tiene dos autores.

`\titulacion{Texto}` Este campo representa el nombre de la titulación que has cursado y de la cual presentarás la memoria de tu proyecto. En nuestro caso serían: Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas.

`\tutor{Nombre}` Dentro de este campo *Nombre* recoge el nombre completo del tutor de tu proyecto de fin de carrera.

`\departamento{Texto}` Este campo contendrá el departamento al que pertenece tu proyecto de fin de carrera.

`\dia{mm/aaaa}` Representa el mes y año en el cual se lleva a cabo la presentación de tu proyecto.

### Comandos

Después del comando `\begin{document}`, cuando inicias propiamente el documento de tu memoria, puedes utilizar los siguientes comandos. Los cuales deberán aparecer en el orden en el que aquí se presentan.

- `\hacerportada` Se encargará de generar la portada de tu memoria.
- `\frontmatter` Marca las secciones que preceden a los capítulos de tu memoria.
- `\input{resumen.tex}` Insertará el contenido de `resumen.tex` en tu memoria, a modo de resumen de la misma.
- `\input{agradecimientos.tex}` Insertará el contenido de `agradecimientos.tex` en tu memoria, generando así la página de agradecimientos.
- `\tableofcontents` Generará el índice general de tu memoria, teniendo en cuenta los capítulos, secciones y subsecciones.
- `\listoffigures` y `\listoftables` Generarán los índices de figuras y cuadros respectivamente.
- `\mainmatter` Marca el inicio del contenido principal de tu memoria, es decir los distintos capítulos que forman parte de ella.
- `\input{Capitulos/capitulo}` Inserta el archivo `capitulo.tex` como uno de los capítulos de la memoria. Tendrás que incluir un comando de estos (y un archivo diferente) por cada capítulo que conforme tu memoria dentro de la carpeta `Capitulos`.
- `\backmatter` Todos los capítulos que insertes después de este comando aparecerán como apéndices en tu memoria.
- `\bibliographystyle{pfcbibstyle}` Indica el estilo de la bibliografía.
- `\bibliography{pfcbib}` Produce, en conjunto con el programa `BibTeX`, la bibliografía de tu memoria. El argumento `pfcbib` indica el archivo `.bib` que contiene los datos de tus referencias bibliográficas.

Finalmente el archivo `proyect.tex` debe terminar con el comando `\end{document}`, el cual indicará la finalización del documento.

## 11.11– Archivos adjuntos a `pclass.cls`

La forma más fácil de instalar es descomprimir todos los archivos contenidos en `pclass.zip` en alguna ubicación donde acostumbres guardar tus propios documentos. Pero debes tener en cuenta un detalle, si necesitas utilizar alguno de los archivos incluidos, alguno de los paquetes por ejemplo, para generar otro documento necesitaras crear copias de esos archivos en las diferentes carpetas según los necesites.

Si prefieres puedes instalar los archivos en el directorio de `LATEX` para que siempre los encuentre y no necesites estar realizando copias innecesarias. En Windows, si estás utilizando `MiKTEX`, debes descomprimir todos

los archivos incluidos en `pclass.zip` dentro del directorio:  
`c:/localtexmf/tex/latex/`<sup>5</sup>. Esto generará una carpeta llamada `pclass` que contiene todos los archivos del paquete.

Dentro de la carpeta `pclass` podrás identificar los archivos que son de ejemplo y puedes moverlos a una nueva carpeta en Mis Documentos o cualquier otra ubicación donde guardes usualmente tus documentos.

Busca ahora, en el Menú Inicio de Windows, los iconos de MikTeX. Has click en el icono *MiKTeX Options* y, en la ventana que aparece, da click en el botón *Refresh Now*. Esto actualiza la base de datos de archivos de L<sup>A</sup>T<sub>E</sub>X para que pueda encontrar el nuevo paquete instalado.

---

<sup>5</sup>Es posible que dentro de `c:/localtexmf/` no encuentres las carpetas `tex/latex/`, si alguna no existe entonces deberás crearla.

---

## Licencia

---

### GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

#### Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

#### GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to

be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipient's exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise)

that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes

make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### END OF TERMS AND CONDITIONS

##### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail. If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY;
for details type 'show w'. This is free software, and you are
welcome to redistribute it under certain conditions; type 'show
c'for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
program
'Gnomovision' (which makes passes at compilers) written by
James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.



---

## Bibliografía

---

- [BC03] Bernardo Cascales, Pascual Lucas, José Manuel Mira Antonio Pallarés y Salvador Sánchez-Pedreño, 2003; *El libro de LATEX*. Prentice Hall, Madrid.
- [Bez07] Bezos, Javier, ‘The titlesec and titletoc packages.’ *Guía propia titlesec*, (2007).
- [Bot97] Botella, Javier Sanguino, 1997; *Iniciación a LaTeX2e*. Addison-Wesley.
- [Cas00] Cascales, Bernardo, 2000; *LaTeX, una imprenta en sus manos*. Aula Documental de Investigación.
- [dS04] de Sousa, José Martínez, 2004.; *Ortografía y ortotipografía del español actual*.
- [Gal92] Gallego, Fernando Ortégón, 1992; *LaTeX primeros pasos*. Editorial Masson.
- [gui] ; *Cada instalación de LATEX debería proporcionar la llamada Guía Local de LATEX, que explica las cosas que son particulares del sistema local. Debería residir en un fichero llamado local.tex. Por desgracia, en algunos sitios no se halla dicha guía. .*
- [Lam94] Lamport, Leslie, 1994; *LaTeX: A document preparation system*. Addison-Wesley.
- [MG94] Michel Goossens, Frank Mittelbach, Alexander Samarin, 1994; *The LaTeX Companion*. Addison-Wesley.
- [Val97] Valiente, Gabriel, 1997; *Composición de textos científicos con LaTeX*. Edicions UPC, Barcelona.
- [vO04] van Oostrum, Piet, ‘Dept. of computer science utrecht university.’ *Guía propia fancyhdr*, (2004).
- [yPD04] y P.W. Daly, H. Kopka, 2004; *A Guide to LaTeX*. Addison-Wesley Professional.