TeXiS: Una plantilla de LaTeX para Tesis y otros documentos

MANUAL DE USUARIO

Marco Antonio Gómez Martín
Pedro Pablo Gómez Martín

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Noviembre 2009

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

TeXiS: Una plantilla de LaTeX para Tesis y otros documentos

Informe técnico del departamento
Ingeniería del Software e Inteligencia Artificial
IT/2009/3

Versión 1.0+

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Noviembre 2009

Copyright © Marco Antonio y Pedro Pablo Gómez Martín

ISBN 978-84-692-7109-4

Al duque de Béjar
y
a tí, lector carísimo

I can't go to a restaurant and
order food because I keep looking
at the fonts on the menu.
Donald Knuth

Agradecimientos

A todos los que la presente vieren y entendieren.

Inicio de las Leyes Orgánicas. Juan Carlos I

Groucho Marx decía que encontraba a la televisión muy educativa porque cada vez que alguien la encendía, él se iba a otra habitación a leer un libro. Utilizando un esquema similar, nosotros queremos agradecer al Word de Microsoft el habernos forzado a utilizar LaTeX. Cualquiera que haya intentado escribir un documento de más de 150 páginas con esta aplicación entenderá a qué nos referimos. Y lo decimos porque nuestra andadura con LaTeX comenzó, precisamente, después de escribir un documento de algo más de 200 páginas. Una vez terminado decidimos que nunca más pasaríamos por ahí. Y entonces caímos en LaTeX.

Es muy posible que hubiéramos llegado al mismo sitio de todas formas, ya que en el mundo académico a la hora de escribir artículos y contribuciones a congresos lo más extendido es LaTeX. Sin embargo, también es cierto que cuando intentas escribir un documento grande en LaTeX por tu cuenta y riesgo sin un enlace del tipo “*Author instructions*”, se hace cuesta arriba, pues uno no sabe por donde empezar.

Y ahí es donde debemos agradecer tanto a Pablo Gervás como a Miguel Palomino su ayuda. El primero nos ofreció el código fuente de una programación docente que había hecho unos años atrás y que nos sirvió de inspiración (por ejemplo, el fichero `guion do.tex` de TeXIS tiene una estructura casi exacta a la suya e incluso puede que el nombre sea el mismo). El segundo nos dejó husmear en el código fuente de su propia tesis donde, además de otras cosas más interesantes pero menos curiosas, descubrimos que aún hay gente que escribe los acentos españoles con el `\'{\i}`.

No podemos tampoco olvidar a los numerosos autores de los libros y tutoriales de LaTeX que no sólo permiten descargar esos manuales sin coste adicional, sino que también dejan disponible el código fuente. Estamos pensando en Tobias Oetiker, Hubert Partl, Irene Hyna y Elisabeth Schlegl, autores del famoso “The Not So Short Introduction to LaTeX 2_ε” y en Tomás

Bautista, autor de la traducción al español. De ellos es, entre otras muchas cosas, el entorno `example` utilizado en algunos momentos en este manual.

También estamos en deuda con Joaquín Ataz López, autor del libro “Creación de ficheros LaTeX con GNU Emacs”. Gracias a él dejamos de lado a WinEdt y a Kile, los editores que por entonces utilizábamos en entornos Windows y Linux respectivamente, y nos pasamos a emacs. El tiempo de escritura que nos ahorramos por no mover las manos del teclado para desplazar el cursor o por no tener que escribir `\emph` una y otra vez se lo debemos a él; nuestro ocio y vida social se lo agradecen.

Por último, gracias a toda esa gente creadora de manuales, tutoriales, documentación de paquetes o respuestas en foros que hemos utilizado y seguiremos utilizando en nuestro quehacer como usuarios de LaTeX. Sabéis un montón.

Y para terminar, a Donal Knuth, Leslie Lamport y todos los que hacen y han hecho posible que hoy puedas estar leyendo estas líneas.

Resumen

> *Desocupado lector, sin juramento me podrás creer que quisiera que este libro [...] fuera el más hermoso, el más gallardo y más discreto que pudiera imaginarse.*
>
> Miguel de Cervantes, Don Quijote de la Mancha

TeXiS es un conjunto de ficheros LaTeX que pueden servir para escribir tesis doctorales, trabajos de fin de master, de fin de carrera y otros documentos del mismo estilo. El documento que tienes en tus manos es un manual que explica las distintas características de la plantilla. En los distintos capítulos iremos explicando los ficheros existentes en TeXiS así como su función. También se explican algunas de las características, como por ejemplo ciertos comandos que facilitan la escritura de los documentos.

Aunque el código LaTeX utilizado en TeXiS está muy comentado para su uso fácil, creemos que las explicaciones que aquí se proporcionan pueden ser útiles.

Hay dos distribuciones distintas de TeXiS: el código fuente completo de este manual (de forma que TeXiS es "*su propio manual*"[1]), o una distribución casi "vacía de contenido", que tiene un único capítulo y apéndice vacío, pero mantiene la portada, dedicatoria, agradecimientos y bibliografía del manual.

Dependiendo, pues, de qué distribución escojas, partirás directamente de los ficheros `.tex` de este manual y eliminarás su texto para añadir el tuyo, o de un conjunto de ficheros sin apenas contenido que rellenarás. Aconsejamos esta última aproximación por ser más cómoda. Sin embargo, hacemos disponible los ficheros `.tex` del manual como referencia.

Para facilitar las cosas, hemos intentado que su estructura sea parecida a la de una posible tesis. De esta forma el código fuente del propio manual puede servir como punto de partida para la escritura de este tipo de documentos. Como podrás comprobar, en algún momento nos ha sido difícil justificar la existencia de ciertos elementos pues no eran realmente relevantes para

---

[1] Los expertos en lógica seguro que tendrían algo que decir al respecto...

el manual. En esos casos, piensa que están ahí no porque sean importantes desde el punto de vista de *este* documento, sino porque muy posiblemente estarían en el tipo de textos para los que TeXiS es útil.

Al estar compuesto por varios tipos de ficheros, TeXiS se rige por varias licencias:

La plantilla (ficheros en el directorio `TeXiS`) se distribuye bajo la *LaTeX Project Public License* (Licencia Pública del Proyecto LaTeX).

Los ficheros `Makefile` y scripts de apoyo a la generación del documento, se distribuyen bajo licencia GPLv3.

El *manual* de TeXiS se distribuye con una licencia Creative Commons (CC-BY-SA).

Índice

Agradecimientos ix

Resumen xi

I Conceptos básicos **1**

1. Introducción **3**
- 1.1. Introducción 3
- 1.2. Qué es TeXiS 5
- 1.3. Qué no es 6
- 1.4. Estructura de capítulos 6
- Notas bibliográficas 7
- En el próximo capítulo 7

2. Estructura y generación **9**
- 2.1. Estructura de directorios 9
- 2.2. Ficheros con el texto principal del documento 10
- 2.3. Ficheros del documento auxiliares 11
- 2.4. Directorio raiz 12
- 2.5. Ficheros de la plantilla 13
- 2.6. Generando el documento 15
- Notas bibliográficas 16
- En el próximo capítulo 16

3. Proceso de edición **17**
- 3.1. Empezando a escribir 17
- 3.2. Editando el texto 19
  - 3.2.1. Nuevos capítulos (y apéndices) 19
  - 3.2.2. Resumen del capítulo 20
  - 3.2.3. Frases célebres 20
  - 3.2.4. Secciones no numeradas 21

3.2.5. Capítulos especiales 24
3.2.6. Dividiendo el documento en partes 25
3.3. Programando en LaTeX 27
3.4. Modos de generación del documento 27
3.4.1. Comando `com` 28
3.4.2. Comando `comp` 28
3.4.3. Comando `todo` 29
3.5. Acelerando la compilación 30
3.6. Editores de LaTeX y compilación 31
3.7. Control de versiones 32
Notas bibliográficas 33
En el próximo capítulo 34

II Conceptos avanzados 35

4. Gestión de las imágenes 37
4.1. Introducción 37
4.2. Gestión de imágenes 38
4.3. Formato de las imágenes 40
4.4. Imágenes independientes del programa generador 41
4.5. Gestión de imágenes y control de versiones 41
4.6. Imágenes divididas 42
Notas bibliográficas 46
En el próximo capítulo 46

5. Bibliografía y acrónimos 47
5.1. Bibliografía 47
5.1.1. Ficheros involucrados 48
5.1.2. Referencias con `natbib` 48
5.1.3. Modificaciones en los `@bibitem` 49
5.1.4. Cambio del estilo de la bibliografía 51
5.2. Acrónimos 52
5.2.1. Acrónimos con glosstex 52
5.2.2. Acrónimos en TeXiS 56
5.2.3. Más allá de TeXiS 57
Notas bibliográficas 58
En el próximo capítulo 58

6. Makefile 59
6.1. Introducción 59
6.2. Objetivos del `Makefile` 60

6.3. Funcionamiento interno 62
    6.3.1. La compilación de las imágenes 62
    6.3.2. Makefile, GlossTEX, y cambio de modo de generación . 64
Notas bibliográficas 65

III Apéndices 67

A. Así se hizo... 69
A.1. Edición 69
A.2. Encuadernación 70
A.3. En el día a día 70

Bibliografía 73

Lista de acrónimos 76

Índice de figuras

2.1. Capturas del visor de PDF 15

3.1. Resaltado de secciones en Emacs 24

4.1. Figura utilizada para marcar una imagen por hacer. 39
4.2. Ejemplo de uso de `subfloat`. 45

5.1. Resultado de la lista de acrónimos 52

A.1. Encuadernación y márgenes guillotinados 71
A.2. Servidor de integración contínua 72

Índice de Tablas

3.1. Secciones no numeradas soportadas por TeXiS 23

4.1. Formatos de imágenes para `latex` y `pdflatex` 40

5.1. Distintas opciones de referencias con `natbib` 49

Parte I

Conceptos básicos

Esta primera parte del manual presenta los conceptos básicos de TeXiS. Contiene un capítulo de introducción, seguido de una descripción de la estructura de TeXiS y cómo se genera el documento final, para terminar con un capítulo en el que se describe el proceso de edición sugerido y los comandos que TeXiS proporciona para facilitar dicho proceso.

En realidad la división por partes del manual no aporta demasiado al lector; se ha dividido en varias partes debido a que, en la práctica, el código de este manual sirve como ejemplo de uso de TeXiS.

En un contexto distinto, es posible que un manual de este tipo no habría tenido estas partes así de diferenciadas.

Capítulo 1

Introducción

> *Púsose don Quijote delante de dicho carro, y haciendo en su fantasía uno de los más desvariados discursos que jamás había hecho, dijo en alta voz:*
>
> Alonso Fernández de Avellaneda, El Ingenioso Hidalgo Don Quijote de la Mancha

Resumen: Este capítulo presenta una breve introducción a TeXiS. El lector podrá hacerse una idea de qué es y para qué sirve. También se encuentra aquí una descripción del resto de capítulos del manual.

1.1. Introducción

Si estás leyendo estas líneas es muy posible que haya llegado la hora de ponerte a escribir la tesis, después de mucho tiempo dando vueltas al área de investigación concreta en el que estás inmerso. O puede que estés a punto de empezar a escribir la memoria del proyecto de fin de carrera, fin de master, o cualquier otro documento de cierta envergadura.

Sea lo que sea lo que te traes entre manos, lo más probable es que no sea fácil hacerlo. Muy posiblemente no tengas aún muy claro qué vas a escribir, pero tu tutor/director/profesor te ha dicho que vayas empezando a plasmar esas ideas sobre el papel para tener algo firme, y sentir que vas avanzando.

Y entonces viene el problema de cómo escribirlo. Muy posiblemente habrás escrito algún artículo en LaTeX y estés convencido de que esa es la vía a seguir para hacer un documento que superará las 10 páginas y que tendrá bibliografía. O puede, simplemente, que alguien te haya dicho que lo mejor es que escribas el proyecto en LaTeX porque la apariencia final es mejor, porque es más cómodo, o cualquier otra razón.

Sea como fuere, parece que estás más o menos decidido a escribir tu documento en LaTeX. Bien hecho. Pero, ¿cómo?. Al contrario de lo que suele ocurrir en congresos y en revistas, no tienes disponible ninguna página en la que descargarte las “instrucciones para los autores”, con la cómoda plantilla en LaTeX que tú, sufrido autor, simplemente tienes que rellenar. No. Ahora las cosas son más complicadas.

Así que te vas a la guía de LaTeX con la que empezaste (apostamos que es la misma con la que hemos empezado todos), y ves las distintas posibilidades que te ofrece en su “`documentclass`”: `article`, `report`, `book`, ... Y te quedas con la última. Pero te asaltan muchas preguntas. ¿Cómo organizo todo esto? o ¿cómo hago la portada? o incluso ¿qué hago para que no ponga “Chapter”, sino “Capítulo”?. En ese punto, es de suponer, has pedido ayuda a la gente de alrededor y/o a tu buscador de Internet favorito. Y de alguna forma, te has encontrado leyendo estas líneas.

Tenemos que decir que exactamente esa fue nuestra situación cuando por fin nos decidimos a escribir nuestras tesis. Desgraciadamente, ni la gente que teníamos alrededor ni nuestro buscador favorito supieron contestarnos de forma satisfactoria, por lo que tuvimos que invertir *mucho tiempo* hasta conseguir que el resultado que salía de nuestros `.tex` nos gustara, hasta que nos sentimos cómodos con la estructura de los ficheros, con las macros disponibles y con el modo de compilación.

Y para que nadie más pueda utilizar como excusa el no saber cómo personalizar la clase `book` para retrasar el comienzo de su tesis, para que nadie más se decida por Word u otro paquete ofimático en vez de LaTeX porque lo ve mucho más sencillo, en definitiva, para que nadie pierda tanto tiempo como perdimos nosotros creando la estructura, decidimos hacer público el esqueleto básico que construimos nosotros para hacerlas. Ese esqueleto básico o plantilla es TeXiS.

En vez de hacer disponible la plantilla o ficheros `.tex` sin ningún contenido, proporcionamos un manual en formato PDF que (a no ser que estés leyendo directamente el código LaTeX), será lo que estás leyendo. Este manual ha sido creado *con la propia plantilla*. Por lo tanto, la distribución de TeXiS es en realidad el código fuente de *su propio manual*. Con su código fuente entre tus manos, lo único que tienes que hacer es borrar su contenido (*este texto*), y rellenarlo con tu gran contribución al mundo. Como podrás comprobar, la estructura del propio manual sigue el esquema de lo que podría ser una tesis, trabajo de investigación o proyecto de fin de carrera, precisamente para que sea fácil quitar el contenido textual y sustituirlo por el nuevo.

En los capítulos que siguen encontrarás toda la información necesaria para poder utilizar los ficheros LaTeX para crear tus propios documentos. Además, el propio código fuente está lleno de comentarios (especialmente en los ficheros que definen el estilo), por lo que también en ellos encontrarás una buena fuente de información. Eso es especialmente importante en caso de que quieras modificar en algo el aspecto final de tu documento.

Esperemos que te sea de utilidad. Si es así, nos gustaría que lo reconocieras en la sección de agradecimientos. Si durante tu proceso de escritura has añadido algún aspecto que crees que puede ser interesante para otros, no dudes en decírnoslo para intentar incluirlo en siguientes versiones de la propia plantilla; tampoco dudes en enviarnos sugerencias sobre las explicaciones de este manual para poder mejorarlo con el tiempo. Por último, también puedes enviarnos el resultado final para poner una referencia a él en la página de descarga, donde, por cierto, puedes ver otros documentos creados con la plantilla, lo que te permitirá coger ideas de cosas que puedes variar. Recuerda que la versión más reciente de TeXIS está disponible en `http://gaia.fdi.ucm.es/projects/texis/`.

1.2. Qué es TeXIS

La plantilla que tienes entre las manos es, como hemos dicho, el esqueleto del código fuente de las Tesis Doctorales de los dos autores (Gómez-Martín, 2008a,b). Por tanto, sirve para escribir otras Tesis Doctorales u otros documentos con estructura similar de forma fácil.

TeXIS te permite además generar el fichero utilizando tanto el comando `latex` (que genera de forma nativa ficheros `dvi` que luego se convierten a ficheros `ps` o `pdf`), como `pdflatex`. De esta forma el usuario final puede elegir entre cualquiera de las dos herramientas[1]. Aconsejamos, no obstante, la utilización de este último, debido a que TeXIS contiene ciertos comandos para dotar al PDF final de marcadores que permiten una navegación cómoda por el fichero utilizando los visores tradicionales.

Como explicaremos en el capítulo siguiente, la plantilla se aprovecha mejor en sistemas GNU/Linux. Nota que hemos dicho que la plantilla “*se aprovecha mejor*” en sistemas GNU/Linux, no que *no pueda utilizarse* en Windows o Mac; es evidente que LaTeX es multiplataforma, y por lo tanto puede compilarse en cualquier sistema que tenga instalada una distribución del mismo.

La razón por esta “desviación positiva” hacia Linux estriba en que para hacer más cómodo el proceso de edición y compilación, TeXIS proporciona ficheros que facilitan el proceso de generación del fichero PDF final, tal y como se describe en el capítulo 6. Esos ficheros adicionales sólo funcionan correctamente si son ejecutados en Linux.

---

[1] Esto es útil por ejemplo cuando quieres utilizar `pdflatex` pero finalmente el servicio de publicaciones sólo admite el uso de `latex`.

1.3. Qué no es

Esta plantilla *no* es un manual de LaTeX, ni una guía de referencia, ni un compendio de preguntas frecuentes. De hecho, no nos consideramos expertos en LaTeX, por lo que no tendríamos fuerzas para escribir algo así. Si necesitas un manual de LaTeX, puedes encontrar muchos y muy buenos en Internet. Al final de este capítulo aparece una lista con algunos de ellos.

La plantilla tampoco es *una clase* de LaTeX. Si miras el código fuente podrás comprobar que el documento comienza con `\documentclass{book}`[2], por lo que se basa en la clase `book`.

La plantilla tampoco te ayudará a gestionar tu bibliografía. Los `.bib` los tendrás que crear y organizar tú ya sea de forma manual o con alguna herramienta diseñada para ello.

Queremos una vez más insistir antes de terminar que no somos expertos en LaTeX. Durante el proceso de escritura de nuestras Tesis nos tuvimos que enfrentar a problemas de formato que tuvimos que solucionar buscando en Internet o preguntando a personas cercanas. Y podemos decir que prácticamente todos los problemas a los que nos hemos enfrentado en nuestra vida como usuarios de LaTeX están resueltos aquí, pues esas Tesis han sido los documentos más extensos que hemos escrito.

Por lo tanto, si tienes alguna duda concreta de LaTeX, en vez de preguntarnos a nosotros, busca en foros de Internet o en la documentación del paquete que estás utilizando. A buen seguro encontrarás ahí la respuesta. Si la duda que tienes es relativa a la plantilla, revisa los comentarios que encontrarás en el código fuente, hay ciertas cosas de demasiado bajo nivel que hemos preferido no contar en el texto. Y sólo como último recurso, preguntanos a nosotros, aunque ya te advertimos que puede que no sepamos responderte. Querríamos poder animarte a escribirnos tus dudas, pero preferimos no hacerlo para no decepcionarte.

1.4. Estructura de capítulos

El manual está estructurado en los siguientes capítulos:

- El capítulo 2 describe a vista de pájaro los distintos ficheros que forman TeXiS. Además da una primera aproximación a cómo generar el documento final (`.pdf`).
- El capítulo 3 se centra en el proceso de edición. Aunque aparentemente la tarea de escribir el texto es trivial, TeXiS proporciona una serie de comandos que pueden ser útiles durante la escritura (al menos a

---

[2] Personalizado, eso sí, para que utilice DIN A-4, a doble cara y con letra de 11 puntos.

nosotros nos lo parecieron). Este capítulo se centra en la explicación de esos comandos.

- El capítulo 4 pasa a describir cómo se estructuran las imágenes en TeXiS. Igual que antes, esto puede parecer superfluo a un usuario medio de LaTeX, pero TeXiS contiene algunos comandos que esperan esa estructura. Es el usuario el último que decide si utiliza esos comandos (y por lo tanto esa estructura) u opta por otra completamente distinta.
- El capítulo 5 aborda la bibliografía y la gestión de los acrónimos. Como se verá, TeXiS dispone de algunas opciones de personalización que merecen un pequeño capítulo.
- El capítulo 6 pone fin al manual, detallando las opciones del fichero `Makefile` que permiten una generación cómoda del documento final en entornos Linux.

El manual tiene, por último, un apéndice que, si bien no es interesante desde el punto de vista del usuario, nos sirve de excusa para proporcionar el código LaTeX necesario para su creación: a modo de “así se hizo”, comenta brevemente cómo fue el proceso de escritura de nuestras tesis.

Notas bibliográficas

El “libro” por el que la mayoría de la gente empieza sus andaduras con LaTeX es Bautista et al. (1998) pues es relativamente corto, fácil de leer y de acceso público (licencia GPL), por lo que se puede conseguir la versión electrónica fácilmente. Un libro algo más completo que éste y que suele ser el segundo en orden de preferencia es Oetiker et al. (1996) con la misma licencia. Dentro de los libros dedicados a LaTeX de libre distribución, también se puede contar con Krishnan (2003).

No obstante, los libros de LaTeX más conocidos son “The LaTeX Companion” (Mittelbach et al., 2004) y “LaTeX: A Document Preparation System” (Lamport, 1994).

En el próximo capítulo...

Una vez hecha una descripción de TeXiS, el próximo capítulo describe los ficheros que componen tanto la plantilla como el manual que estás leyendo. También se explicará cómo se puede generar o compilar el manual a partir de los `.tex` proporcionados. Por lo tanto, el capítulo sirve como una primera aproximación rápida al trabajo con TeXiS; al final del mismo seremos capaces de entender la estructura de directorios propuesta y dónde se encuentran los ficheros que hay que editar para cambiar el contenido del documento final.

No obstante, el capítulo siguiente debe verse únicamente como una primera aproximación. El capítulo 3 da más detalles sobre el proceso de edición del documento, y el capítulo 6 dará una alternativa al modo de compilación explicado.

Capítulo 2

Estructura y generación

> *La mejor estructura no garantizará los resultados ni el rendimiento. Pero la estructura equivocada es una garantía de fracaso.*
>
> Peter Drucker

Resumen: Este capítulo explica la estructura de directorios de TeXIS así como los ficheros más importantes, describiendo el cometido de cada uno. También hace una primera aproximación al proceso de generación (o compilación) del PDF final, aunque este tema será extendido posteriormente en el capítulo 6.

2.1. Estructura de directorios

Como habrás podido comprobar, la plantilla contiene bastantes ficheros organizados en varios directorios. Esta sección explica el contenido de cada uno de los directorios, para que seas capaz de encontrar el directorio en el que debería estar un fichero concreto.

Existen los siguientes directorios:

Directorio raíz contiene el fichero principal del documento (también llamado fichero *maestro*), que es el que se utiliza como entrada a `pdflatex` (o `latex`) y cuyo nombre es `Tesis.tex`. También aparecen en el directorio otros ficheros que si bien no generan texto en el documento final cumplen ciertas funciones específicas descritas en la sección 2.4. Por último, el directorio contiene también los ficheros `.bib` con la información bibliográfica así como el fichero para generar el documento utilizando la aplicación `make`.

Directorio `./Capitulos` contiene los `.tex` de cada capítulo del documento.

Directorio `./Apendices` contiene los `.tex` de cada uno de los apéndices.

Directorio `./Cascaras` contiene los `.tex` responsables del contenido del resto de páginas del documento: el texto de la portada, agradecimientos, resumen, etc. En definitiva son los ficheros responsables de todo aquello que precede a los capítulos y sigue a los apéndices.

Directorio `./Imagenes` contiene las imágenes del documento. Dentro de él aparecen varios directorios distintos. La gestión de imágenes (y por lo tanto la estructura de estos directorios) se describirá en el capítulo 4.

Directorio `./TeXiS` contiene todos los ficheros relacionados con la propia plantilla, es decir, los ficheros que definen la apariencia final del documento, así como los comandos que facilitan la edición que serán descritos en el capítulo 3. La creación de un documento que se adhiere completamente al formato de TeXiS no necesitará tocar ninguno de los ficheros de este directorio.

Directorio `./VersionesPrevias` Este directorio es usado por el `Makefile` cuando se realiza una copia de seguridad del estado del documento. Describiremos esta característica en el capítulo 6.

Existen por lo tanto, tres tipos de ficheros `.tex`: los ficheros que contienen el texto principal del documento (capítulos y apéndices), los ficheros que definen las partes adicionales del mismo (como portada y agradecimientos), y los ficheros que determinan la apariencia. En las tres secciones siguientes describimos cada uno de ellos.

2.2. Ficheros con el texto principal del documento

Estos `.tex` son los que contienen el texto tanto de los capítulos como de los apéndices, por lo tanto son los ficheros que más tiempo pasarás editando. Están divididos en secciones, tienen figuras, tablas, referencias bibliográficas, y cualquier otro tipo de elemento que quieras o debas añadir.

En principio pueden contener cualquier código LaTeX. No obstante, no olvides que si necesitas algún paquete especial que no se cargue por defecto en la plantilla, deberás incluir el `\usepackage` correspondiente en el documento maestro o en el fichero de preámbulo de TeXiS, `TeXiS/TeXiS_pream.tex` descrito en la Sección 2.5.

El capítulo siguiente está enteramente dedicado al proceso de edición de estos ficheros.

2.3. Ficheros del documento auxiliares: las cáscaras del documento

Estos ficheros, como ya hemos dicho, son los responsables del contenido del resto de páginas del documento, todo aquello que no son capítulos o apéndices. Son los siguientes (por orden de "aparición" en el documento final)[1]:

- `cover.tex`: responsable de las dos primeras hojas del documento, que forman las portada. Mediante comandos se definen el autor y título que aparecerá en la portada, la fecha de publicación, facultad, etc. Como podrás ver cuando lo edites, el fichero contiene los datos concretos para generar este manual. Los comandos se describen en la sección 3.1.

- `dedicatoria.tex`: contiene el código LaTeX que crea la "dedicatoria" de la Tesis. Consiste en una hoja donde aparece alineada a la izquierda una frase indicando a quién se "dedica" el documento (en los libros serios pone algo como "A mis padres", aunque también hay autores en libros más distendidos, como Buckland que dice textualmente "For Mum and Dad, who bought me my first computer, and therefore must share some responisibility for turning me into the geek that I am" (Buckland, 2005)). Se pueden poner todas las páginas de dedicatorias que se deseen, utilizando la macro `\putDedicatoria`, que recibe la cita completa y crea la hoja completa con la misma. Lo más cómodo, no obstante, es utilizar la macro `\dedicatoriaUno` y (opcionalmente) `\dedicatoriaDos` para establecer las dos dedicatorias y a continuación invocar `\makeDedicatorias` para generarlas. Así lo hace este manual.

- `agradecimientos.tex`: contiene el texto de las únicas páginas que tu familia y amigos van a leer de la Tesis: los agradecimientos. Así que piensa bien lo que pones, no olvides a nadie[2].

  Es importante que no borres la línea que aparece justo después del `\chapter`,

  ```
  \cabeceraEspecial{Agradecimientos}
  ```

  ya que lo que hace es modificar la cabecera de la página para que no aparezca con el mismo formato que en los capítulos. Puedes consultar la sección 3.2.5 para obtener más detalles sobre esto.

---

[1] Si crees que no necesitas alguno de ellos, puedes eliminar su inclusión en el fichero maestro, `Tesis.tex`.

[2] Tampoco a nosotros por quitarte la preocupación del aspecto final... :-)

- `resumen.tex`: si quieres incluir antes del índice un pequeño resumen de tu trabajo, puedes hacerlo en este fichero. Al igual que en los agradecimientos no debes eliminar el comando LaTeX del principio que altera la cabecera.

  Tanto el resumen como los agradecimientos antes explicados se convierten en dos "capítulos sin numeración" que también serán listados en el índice de contenidos. No obstante, al aparecer antes que el texto principal del documento (los capítulos propiamente dichos), sus páginas serán numeradas con notación romana, en lugar de con la arábiga tradicional.

- `bibliografia.tex`: en él se configura la bibliografía del documento. En concreto, el fichero permite indicar tanto qué ficheros `.bib` contienen las entradas bibliográficas como una frase célebre (seguramente, ya habrás notado que TeXiS permite iniciar los capítulos con una frase célebre), característica descrita con más detalle en la sección 3.2.3.

  El capítulo 5 hace una descripción más detallada del tipo de bibliografía que propone utilizar la plantilla (y que utiliza este manual).

- `fin.tex`: En nuestras respectivas tesis, como "cierre" incluimos una última página parecida a la dedicatoria con un par de frases célebres. El código TeX responsable se encuentra en este fichero.

Existen otros dos ficheros que no aparecen en este directorio pero que generan páginas en el documento final. Son `TeXiS_toc.tex` y `TeXiS_acron.tex` del directorio `TeXiS`, descritos en la sección 2.5. Aparecen en ese directorio debido a que no permiten ningún tipo de personalización al usuario de TeXiS.

2.4. Directorio raiz

En el directorio raíz aparecen, además de `Tesis.tex`, el documento maestro, otros tres ficheros `.tex` que no son responsables de la generación de ninguna página del documento. Uno de ellos, `config.tex` se describe en la sección 3.4. Los otros dos son:

- `guionado.tex`: contiene una lista de aquellas palabras que, durante la edición del documento, se ha podido comprobar que LaTeX dividía mal. En esos casos, la alternativa mala es hacer pequeños ajustes en el párrafo para que esa palabra cuyos guiones LaTeX no sabe colocar no quede cerca del final de la línea. La alternativa buena es añadir la palabra a este fichero, colocando los guiones donde van. En el fichero proporcionado aparece una lista de algunas palabras de ejemplo.

- `constantes.tex`: está pensado para la definición de constantes que aparezcan a menudo en el texto. Por ejemplo, si se hace un documento sobre Cruise Control (ThoughtWorks, 2001), para evitar tener que escribir contínuamente las dos palabras, es buena idea incluir una constante en el fichero que cree un comando para hacerlo más rápidamente:

  ```
  \newcommand{\cc}{Cruise Control}
  La nueva versión de \cc\ \ldots
  ```

  La nueva versión de Cruise Control ...

  En este fichero aparece definida la constante `\titulo` que contiene el título del documento y `\autor` con el autor. Ambos son utilizados en la portada. También aparece definido el comando `\texis` que utilizamos en este manual para evitarnos escribir el código que escribe “TeXiS” una y otra vez:

  ```
  \texis\ te permite generar el
  fichero final tanto como .dvi
  como en un .pdf.
  ```

  TeXiS te permite generar el fichero final tanto como .dvi como en un .pdf.

Por último indicar que en el directorio raíz aparecen los ficheros con extensión `.bib` que contienen la información bibliográfica y los `.gdf` para los acrónimos (ver capítulo 5) así como el fichero `Makefile` para la generación automática del documento final (capítulo 6).

2.5. Ficheros de la plantilla

El directorio `TeXiS` contiene los ficheros que definen la apariencia final del documento. Si el formato de este manual te gusta tal cual, no tendrás por qué tocar ninguno de estos ficheros. La explicación de su contenido aparece a continuación. Su código fuente contiene numerosos comentarios y enlaces, por lo que no debería suponerte demasiado problema modificarlos.

- `TeXiS_cab.tex`: contiene la definición de la apariencia de las cabeceras de las páginas utilizadas en el documento. La plantilla utiliza el paquete `fancyhdr`. Sin embargo, la cabecera por defecto se ha modificado para que aparezca el número del capítulo, así como su nombre en minúsculas, junto con algún otro cambio menor.

- `TeXiS.sty`: contiene los comandos que la plantilla proporciona para facilitar el proceso de edición. El uso de estos comandos está explicado en el capítulo 3. A pesar de que la extensión distinta a la habitual (`.sty` en vez de `.tex`) puede imponer cierto respeto al principio, puedes abrir sin miedo el fichero para edición, pues es un fichero de LaTeX normal, con definiciones de comandos tradicionales.

- `TeXiS.bst`: contiene el estilo que utiliza la plantilla para generar la lista de las referencias bibliográficas al final del documento. Las posibilidades de este estilo son descritas en el capítulo 5.

- `TeXiS_pream.tex`: este fichero contiene la mayor parte del código del preámbulo del documento (lo que va antes del `\begin{document}`). En él aparecen incluidos un buen número de paquetes que pueden ser útiles en la elaboración del documento, junto con una explicación de para qué sirven y, en algunas ocasiones, algunos ejemplos de uso. Existen incluso ciertos paquetes cuya inclusión aparece comentada pero que se mantienen, junto con su comentario correspondiente, por si pueden venir bien para documentos concretos que necesiten ciertas características que ni este manual ni nuestras tesis requirieron.

- `TeXiS_cover.tex`: contiene el código TeX que genera la portada, y la hoja siguiente a la misma, que vuelve a tener los mismos datos pero sin el escudo.

- `TeXiS_dedic.tex`: contiene el códito TeX para generar las hojas de dedicatorias.

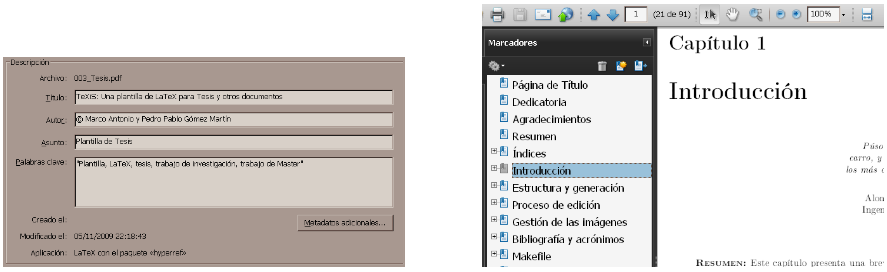
- `TeXiS_toc.tex`: es el responsable de la generación de los índices de capítulos, tablas y figuras que aparecen en el documento.

- `TeXiS_bib.tex`: es el encargado de que en el documento aparezca bibliografía. Incluido desde el fichero maestro, lo primero que hace es leer el fichero de configuración, `Cascaras/configBibliografia.tex`.

  Como puedes comprobar, la bibliografía es también referenciada en el índice como un capítulo sin numerar; también se preocupa de cambiar la cabecera para que no se utilice la habitual del resto de capítulos.

- `TeXiS_acron.tex`: la plantilla también permite añadir una lista de acrónimos o abreviaturas utilizadas en el texto. En este fichero se incluyen los comandos necesarios para que aparezca esta lista. No obstante, para que la lista funcione, en el momento de la generación se debe invocar a la herramienta correspondiente para que se creen los ficheros auxiliares necesarios para su generación. En la descripción sobre la generación dada en la sección 2.6 no se describe este proceso, por lo que el resultado contendrá una lista de acrónimos vacía. El uso de acrónimos se describe con detalle en la sección 5.2.

- `TeXiS_part.tex`: contiene los comandos relacionados con la posibilidad de dividir en *partes* el documento final. Los detalles de qué posibilidades ofrece TeXiS para hacerlo están descritas en la sección 3.2.6.



Descripción
Archivo: 003_Tesis.pdf
Título: TeXiS: Una plantilla de LaTeX para Tesis y otros documentos
Autor: © Marco Antonio y Pedro Pablo Gómez Martín
Asunto: Plantilla de Tesis
Palabras clave: "Plantilla, LaTeX, tesis, trabajo de investigación, trabajo de Master"
Creado el:
Modificado el: 05/11/2009 22:18:43
Aplicación: LaTeX con el paquete «hyperref»

Metadatos adicionales...
1
(21 de 91)
100%
Marcadores
Página de Título
Dedicatoria
Agradecimientos
Resumen
Índices
Introducción
Estructura y generación
Proceso de edición
Gestión de las imágenes
Bibliografía y acrónimos
Makefile
Capítulo 1
Introducción

(a) Propiedades del documento (b) Tabla de contenidos

Figura 2.1: Capturas del visor de PDF

2.6. Generando el documento

Como ya se dijo en la introducción, TeXiS permite compilar el documento[3] tanto con `latex` como `pdflatex`. Si has utilizado LaTeX a través de editores de texto específicos (como Kile o WinEdt), es posible que no sepas de qué estamos hablando. Tanto `latex` como `pdflatex` son dos aplicaciones que cogen un fichero `.tex` como entrada y generan el documento final “renderizado”. La diferencia entre ambas radica en el fichero de salida que generan. En el primer caso, se genera un fichero `.dvi`[4], mientras que en el segundo caso se genera un fichero PDF directamente. Tradicionalmente se ha utilizado `latex`, convirtiendo después el fichero `.dvi` al formato deseado (como `.ps` o `.pdf`). Sin embargo, en nuestro caso, aconsejamos la utilización de `pdflatex`, debido a que, al generar de forma nativa ficheros PDF, aprovecha algunas de las características disponibles en los mismos. En particular, TeXiS contiene algunos comandos LaTeX que `pdflatex` aprovecha para añadir información de *copyright* al fichero, así como enlaces a cada uno de los capítulos y secciones del documento, permitiendo una navegación rápida por el mismo cuando se utilizan visores (figura 2.1).

La plantilla incluye un fichero `Makefile` para automatizar la generación del fichero final[5] que es capaz de crear el PDF utilizando cualquiera de las dos alternativas. No obstante, en este apartado no entraremos en los detalles de este fichero, ya que existe un capítulo dedicado enteramente a él (capítulo 6).

Para generar el documento de este manual a partir de los ficheros de TeXiS proporcionados, la forma inmediata es seguir el proceso tradicional de generación de cualquier fichero de LaTeX, es decir, ejecutar `pdflatex` (o `latex`), a continuación ejecutar `bibtex` para resolver las referencias bibliográficas, y posteriormente ejecutar un par de veces más `pdflatex` para resolver las referencias cruzadas y que aparezcan en el documento final.

---

[3] Cuando hablamos de “compilación” nos referimos, por analogía con el desarrollo software, a la generación del fichero final (un PDF) resultado de analizar los ficheros fuente en LaTeX.

[4] *Device independent*, o “independiente del dispositivo” (en el que se mostrará el contenido).

[5] Los ficheros `Makefile` son ampliamente utilizados en el desarrollo de software. Son ficheros que sirven de entrada a la utilidad `make` que genera automáticamente los ficheros de resultado a partir de los archivos de código fuente.

En línea de comandos esto se traduce a las siguientes órdenes[6]:

```
$ pdflatex Tesis
$ bibtex Tesis
$ pdflatex Tesis
$ pdflatex Tesis
```

Si se utiliza algún editor de LaTeX para la edición, también se pueden utilizar sus teclas rápidas (o en su defecto, sus botones u opciones de menú) para generarlo; encontrarás una explicación al respecto en la sección 3.6.

Notas bibliográficas

En este capítulo hemos descrito simplemente la estructura de directorio de TeXiS, por lo que no existe ninguna fuente relacionada adicional de consulta. Se mantiene este apartado por simetría con el resto de capítulos. En un documento normal (tesis, trabajo de investigación) lo más probable es que todos los capítulos puedan extenderse con notas de este tipo.

En el próximo capítulo...

Una vez que se han descrito a vista de pájaro los ficheros que componen la plantilla y una primera aproximación al proceso de generación del documento final (en PDF), el siguiente capítulo pasa a describir el proceso de edición.

Eso cubre aspectos tales como los ficheros que deben modificarse para añadir nuevos capítulos o los comandos que TeXiS hace disponibles para escribir ciertas partes de los mismos. El capítulo describe también los dos modos de generación del documento final que pueden ser de utilidad durante el largo proceso de escritura. Por último, el capítulo terminará con ciertas consideraciones relativas a los editores de LaTeX utilizados así como sobre la posibilidad de utilizar un control de versiones.

---

[6] También es válido el uso de `latex` en lugar de `pdflatex`, pero el fichero generado (`.dvi`) deberá después ser convertido a PDF.

Capítulo 3

Proceso de edición

Rem tene, verba sequentur (Si dominas
el tema, las palabras vendrán solas)

Catón el Viejo

Resumen: Este capítulo se centra en el proceso de edición, dando detalles de qué cosas deben cambiarse y qué comandos y características tiene TeXiS que facilitan el proceso.

3.1. Empezando a escribir

En primer lugar, es necesario destacar que los ficheros `.tex` *deben tener* codificación ISO-8859-1. Esto es lo que ocurre de manera predefinida en Windows y en algunos Linux como Debian. Una excepción significativa es el caso de Ubuntu, que usa de manera predeterminada UTF-8. En ese caso, deberás ser cuidadoso para asegurarte de que grabas tus ficheros con ISO-8859-1.

El primer paso para la construcción de un nuevo documento es cambiar el título y autores. Es posible que al principio del proceso no se tenga muy claro cuál es el título final del documento pero, y esto es una opinión personal, ver un título (aunque sea provisional) en vez de lo que ahora aparece (“TeXiS: Una plantilla de LaTeX para Tesis y otros documentos”) te ayudará a pensar que lo que estás escribiendo es tuyo y no de otros. Para eso, basta con cambiar la constante `\titulo` y `\autor` que aparece definida en el fichero `constantes.tex`.

El segundo paso es crear la portada en `Cascaras/cover.tex`. Como habrás podido observar, TeXiS genera dos hojas de portada, al igual que hacen la mayoría de los libros. La primera portada es la que iría en la parte exterior del documento encuadernado, mientras que la siguiente es una repetición que aparece en la primera página. A continuación aparece una lista con el texto que puede cambiarse usando los comandos de TeXIS; una vez que se configuran, se debe invocar al comando `\makeCover` para generar las portadas:

- Título del documento: aparece en las dos portadas. Por defecto se utilizará la constante `\titulo` definida en `constantes.tex`. No obstante, se puede indicar un título distinto usando `\tituloPortada`. De esta forma, se pueden forzar saltos de línea artificiales si se desea.
- Autor del documento: normalmente aparece también en las dos portadas. Igual que antes, si no se indica lo contrario se utiliza `\autor`, aunque se puede cambiar con `\autorPortada`.
- Una imagen en la primera portada, normalmente el escudo institucional. El fichero a utilizar se define con `\imagenPortada`. También puede especificarse la escala a utilizar en el fichero si éste es demasiado grande o pequeño con `\escalaImagenPortada`.
- Una fecha de publicación, que aparece en la parte inferior de ambas portadas. Se utiliza el comando `\fechaPublicacion`.
- El "tipo de documento" que aparece en la primera portada. Si no se indica nada, será "TESIS DOCTORAL". Se puede modificar con `\tipoDocumento`. Este manual por ejemplo lo establece en "MANUAL DE USUARIO".
- El departamento y facultad al que está asociado el documento. Aparece en ambas portadas, y se establece con `\institucion`.
- Un primer bloque de texto en la segunda portada, que aparece después del título. Si no se indica lo contrario, en ese bloque aparecerá el texto "Memoria que presenta para optar al título de Doctor en Informática" seguido del `\autorPortada`. Se puede cambiar el contenido completo con `\textoPrimerSubtituloPortada`.
- Un segundo bloque de texto donde aparece "Dirigida por el Doctor" seguido del director del trabajo que se establece con `\directorPortada`. El comando `\textoSegundoSubtituloPortada` permite establecer otro texto distinto.

Las dos portadas en sus caras traseras pueden, además, presentar otra información auxiliar:

- Un breve recordatorio indicando que el documento está preparado para su impresión a doble cara. Si se desea que aparezca, basta con llamar a `\explicacionDobleCara`.

- El ISBN del documento, en caso de poseerlo. Se define con `\isbn`.
- Información de copyright. Se puede indicar con `\copyrightInfo`, y lo habitual será pasar como parámetro el `\autor`.
- Por defecto en la cara posterior de la primera portada aparecen unos “créditos” a TeXiS, donde se indica que el documento se ha generado con TeXiS y la versión. Si no se desea que aparezca, se puede llamar a `\noTeXiSCredits`, aunque nos gustaría que lo incluyeras.

Por último, quizá quieras cambiar la información de “metadatos” que se incrustará en el PDF generado. Los metadatos aparecen directamente en el fichero `Tesis.tex` y, como indicamos en el capítulo anterior y mostramos en la figura 2.1, son:

```
%
% "Metadatos" para el PDF
%
\ifpdf\hypersetup{%
    pdftitle = {\titulo},
    pdfsubject = {Plantilla de Tesis},
    pdfkeywords = {Plantilla, LaTeX, tesis, trabajo de
      investigación, trabajo de Master},
    pdfauthor = {\textcopyright\ \autor},
    pdfcreator = {\LaTeX\ con el paquete \flqq hyperref\frqq},
    pdfproducer = {pdfeTeX-0.\the\pdftexversion\pdftexrevision},
    }
    \pdfinfo{/CreationDate (\today)}
\fi
```

3.2. Editando el texto

Una vez que se tienen el título y autores del documento puestos, el trabajo de escritura consiste, en su mayor parte, en la creación de los correspondientes ficheros LaTeX de cada uno de los capítulos y apéndices.

3.2.1. Nuevos capítulos (y apéndices)

Según la estructura de directorios vista en el capítulo anterior, TeXiS te recomienda crear los capítulos en el directorio `Capitulos` y los apéndices en `Apendices`.

Cuando crees un fichero en cualquiera de los directorios, se debe añadir en el fichero maestro (`Tesis.tex`) el nombre de ese nuevo fichero para que se procese en el momento de la generación:

```
\mainmatter

\include{Capitulos/01Introduccion}
\include{Capitulos/02EstructuraYGeneracion}
...

% Apéndices
\appendix
\include{Apendices/01AsiSeHizo}
...
```

Todos estos ficheros de capítulos y apéndices deben comenzar con el comando LaTeX `\chapter`[1]. El resto del fichero es un fichero LaTeX normal que tendrá secciones, subsecciones, figuras, tablas, etc.

Al añadir un nuevo fichero, es posible que también quieras añadir su nombre en el fichero `config.tex` para permitir la compilación rápida de un único capítulo según se cuenta en la sección 3.5.

3.2.2. Resumen del capítulo

TeXiS permite incluir al comienzo de todos los capítulos un breve resumen del mismo; este mismo manual lo hace. Para separarlo del resto se utiliza un formato distinto.

En vez de cambiar el formato en todos y cada uno de los capítulos (y apéndices), TeXiS proporciona un *entorno* nuevo, `resumen`, que lo hace por nosotros:

```
\begin{resumen}
En este capítulo se describe...
\end{resumen}
```

RESUMEN: En este capítulo se describe...

El formato concreto está definido en el fichero `TeXiS/TeXiS.sty`, por lo que se puede cambiar a voluntad, lo que provocará el cambio en todas sus apariciones.

3.2.3. Frases célebres

Como habrás podido comprobar leyendo este manual, TeXiS permite además escribir en cada capítulo una “frase célebre” que es añadida inmediatamente después del título del mismo, alineada a la derecha.

Para añadir la frase (que está formada por la cita en cuestión y su autor), TeXiS define un nuevo entorno `FraseCelebre`, dentro del cual se especifican cada una de ellas con otros dos entornos, `Frase` y `Fuente`:

---

[1] Esto *también* se cumple para los apéndices.

```
\begin{FraseCelebre}
\begin{Frase}
Nadie espere que yo diga algo.
\end{Frase}
\begin{Fuente}
Mafalda
\end{Fuente}
\end{FraseCelebre}
```

Nadie espere que
yo diga algo.

Mafalda

Evidentemente, las frases célebres pueden añadirse en todos los capítulos, incluidos los “especiales” (aquellos que no tienen numeración normal) como el capítulo de agradecimientos. Para hacerlo, basta con utilizar los comandos anteriores.

Un capítulo donde es algo más complicado es el “capítulo” de *bibliografía*. Esto es debido a que la generación del capítulo completo consiste en una mera invocación al comando `bibliography`

```
\bibliography{fichero1,fichero2}
```

En el `documentclass` que estamos utilizando (`book`) eso significa que se creará un nuevo *capítulo* con la lista de referencias. Si en ese capítulo se quiere añadir una cita (como hacemos por ejemplo en este manual), hay que realizar algunas tareas adicionales. Naturalmente TeXiS las hace por nosotros, por lo que, como se mencionó en la sección 2.3, lo único que tendremos que hacer es editar el fichero `bibliografia.tex`, buscar la frase célebre del manual y cambiarla a voluntad.

Antes de terminar, decir que, igual que en el caso del resumen, la apariencia de la frase célebre se puede modificar en el fichero `TeXiS/TeXiS.sty`.

3.2.4. Secciones no numeradas

Como habrás podido comprobar, en este manual todos los capítulos terminan con dos secciones no numeradas, una de ellas con unas notas bibliográficas, y otra que tiene un pequeño resumen del siguiente capítulo.

Aunque para el manual no son en realidad necesarias (especialmente la de notas bibliográficas, que en muchos capítulos nos ha costado rellenar...), las hemos puesto para que sirvan de ejemplo en el `.tex`.

En principio, para poner una sección no numerada basta con utilizar la “versión estrellada” del comando LaTeX correspondiente. Es decir, utilizar `\section*` para añadir una sección sin número. El problema en nuestro caso es que este comando no parece funcionar correctamente con el paquete `fancyhdr`. TeXiS utiliza ese paquete para configurar la cabecera y pie de página; en concreto para indicar que se desea que el número de página aparezca en las esquinas “externas”, mientras que en las esquinas internas debe aparecer el nombre del capítulo (en las hojas pares o izquierdas) y sección (en las impares o derechas). El mismo paquete es el que se utiliza para que aparezca el número de página en la primera página de un capítulo y para cierta información que aparece cuando se genera el documento en “modo borrador”, según aparece descrito en la sección 3.4.

El problema aparece cuando una sección no numerada excede el límite de la página en la que empieza. En ese caso, la cabecera en la que aparece el nombre de la sección en vez de contener el título de esa sección sin numerar, seguirá mostrando la última sección numerada.

La solución es modificar a mano la cabecera, en concreto modificar la configuración de la cabecera donde aparece el título de la sección actual (la parte izquierda de las páginas impares). Para eso, tras consultar la documentación del paquete, se aprende que hay que utilizar el comando `\markright`. Por ejemplo:

```
\section*{Notas bibliográficas\markright{Notas bibliográficas}}
```

Como puede verse, en el propio comando `\section*`, se incluye una llamada a `\markright`, que contiene el texto que debe aparecer en la cabecera. Con esto se soluciona el problema de las cabeceras.

Otro “problema” de las secciones sin numerar es que no se meten en la tabla de contenidos que se incluye al principio del documento; tampoco aparecen en el “contenido” del PDF listado por el visor que mostrabamos en la figura 2.1[2]. Sin embargo, en nuestro caso preferíamos que también las secciones aparecieran en el índice (es decir, que la única diferencia entre las secciones numeradas y las no numeradas fuera, precisamente, la ausencia de numeración). Para que aparezca, por lo tanto, se debe añadir explícitamente la sección en la tabla de contenidos, con el comando:

```
\addcontentsline{toc}{section}{Notas bibliográficas}
```

que debe ejecutarse *después* del comando `\section*`. Por lo tanto, para añadir una sección sin numerar como la de “Notas bibliográficas”, el código LaTeX final que hay que poner es:

```
%-----------------------------------------------------------------
\section*{Notas bibliográficas\markright{Notas bibliográficas}}
%-----------------------------------------------------------------
\addcontentsline{toc}{section}{Notas bibliográficas}
```

Entendemos que invocar a los comandos anteriores cada vez que se desea una de estas secciones no numeradas es tedioso. Por ello TeXiS proporciona una serie de comandos (definidos en el fichero `./TeXiS/TeXiS_cab.tex`) que permiten añadir fácilmente cuatro tipos de secciones sin numerar. Las secciones son los siguientes (ver tabla 3.1):

---

[2] Ponemos *problema* entre comillas porque normalmente se utiliza la versión con estrella de los comandos `section` precisamente para evitar que una sección aparezca en el índice.

Texto	Comando para `section`	Comando para índice
Conclusiones	`\Conclusiones`	`\TocConclusiones`
En el próximo capítulo...	`\ProximoCapitulo`	`\TocProximoCapitulo`
Notas bibliográficas	`\NotasBibliograficas`	`\TocNotasBibliograficas`
Resumen	`\Resumen`	`\TocResumen`

Tabla 3.1: Secciones no numeradas soportadas por TeXiS

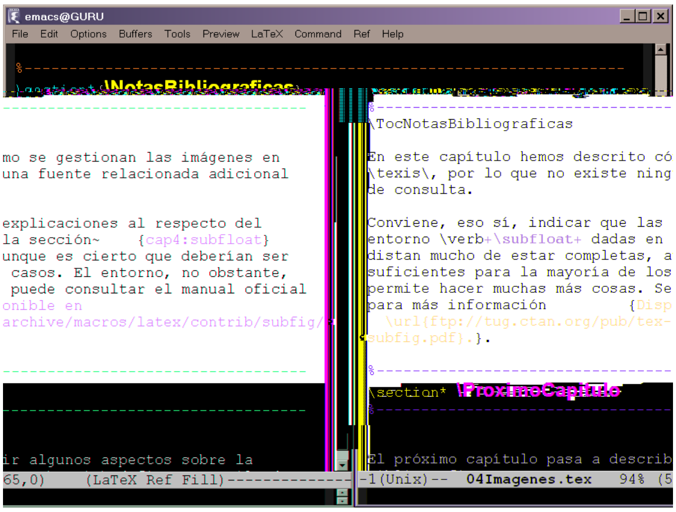
- "Conclusiones": el manual no utiliza esta sección sin numerar, pero sí puede ser razonable utilizarlo a modo de resumen al final del capítulo de otro tipo de documentos.
- "Notas bibliográficas": también utilizado en este documento, es útil para dar otras referencias bibliográficas que por cualquier razón no se citó en el texto.
- "En el próximo capítulo...": sí se ha utilizado en el manual, y puede servir para enlazar el contenido del capítulo con el siguiente.
- "Resumen": con un objetivo parecido al de conclusiones pero con distinto título; tampoco lo utilizamos en el manual.

Como se puede ver en la tabla, para cada una de estas secciones aparecen dos comandos, uno para el comando `\section*` y otro para añadir el índice, de forma que la definición de, por ejemplo, la sección de "En el próximo capítulo..." quedaría:

```
%--------------------------------------------------------------
\section*{\ProximoCapitulo}
%--------------------------------------------------------------
\TocProximoCapitulo
```

Somos conscientes de que los dos comandos podrían haberse unificado en uno sólo, como `\SeccionProximoCapitulo` y que él mismo hiciera todo el trabajo (es decir, pusiera el `\section*{...}` así como el `\addcontestline`). Sin embargo, esta solución no es compatible con la capacidad de los editores de resaltar secciones, ya que los editores simplemente buscan la cadena "`\section`" para resaltarlo (ver figura 3.1).

Es por ello que, a pesar de ser más tedioso, optamos por la alternativa complicada: si se quiere meter una sección sin numerar, se debe primero utilizar el comando `\section*`, añadiendo como texto el comando que aparece en la segunda columna de la tabla 3.1, y posteriormente se utiliza el otro comando para añadirlo al índice. Separándolo así, además, permite al usuario de TeXiS decidir si quiere o no que la sección aparezca en el índice.


emacs@GURU
File Edit Options Buffers Tools Preview LaTeX Command Ref Help

```
%------------------------------------------------------------
\NotasBibliograficas

-----------------------------------

mo se gestionan las imágenes en
una fuente relacionada adicional


explicaciones al respecto del
la sección~    {cap4:subfloat}
unque es cierto que deberían ser
 casos. El entorno, no obstante,
 puede consultar el manual oficial
onible en
archive/macros/latex/contrib/subfig/


-----------------------------------

-----------------------------------


ir algunos aspectos sobre la
65,0)     (LaTeX Ref Fill)-------------
```

```
%-----------------------------------
\TocNotasBibliograficas

En este capítulo hemos descrito có
\texis\, por lo que no existe ning
de consulta.

Conviene, eso sí, indicar que las
entorno \verb+\subfloat+ dadas en
distan mucho de estar completas, a
suficientes para la mayoría de los
permite hacer muchas más cosas. Se
para más información         {Disp
  \url{ftp://tug.ctan.org/pub/tex-
subfig.pdf}.}.

%-----------------------------------
\section* \ProximoCapitulo
%-----------------------------------


El próximo capítulo pasa a describ
-1(Unix)--  04Imagenes.tex   94% (5
```

Figura 3.1: Resaltado de secciones en Emacs

3.2.5. Capítulos especiales

Relacionado con las cabeceras de la sección anterior, TeXIS soporta (y este manual tiene) capítulos "especiales" que aparecen sin numerar. Estos "capítulos" son, en concreto, la parte de agradecimientos y resumen, los índices y la bibliografía.

Dado que todos ellos se caracterizan por la ausencia de secciones, no tiene sentido mantener la cabecera utilizada en el resto del texto. Por lo tanto, configuramos sus cabeceras para que en ambas páginas aparezca directamente el título del capítulo (también sin número).

Para hacerlo, TeXIS dispone del comando `\cabeceraEspecial`, que recibe como parámetro el nombre del capítulo. De esta forma, el capítulo de agradecimientos comienza con:

```
\chapter{Agradecimientos}

\cabeceraEspecial{Agradecimientos}

\begin{FraseCelebre}
...
```

que provoca un cambio en la cabecera que se debe utilizar.

Los capítulos sin numerar de este manual se encargan de configurar la propia cabecera por lo que si partes de ellos para escribir tu documento no deberás preocuparte de nada (más allá de *no* borrar el comando).

Si incluyes nuevos capítulos sin numerar, has de saber que:

- No debes olvidar invocar el comando anterior al principio del capítulo sin numerar.
- El comando anterior *sobreescribe* el funcionamiento normal de la cabecera, por lo que se debe llamar al comando `\restauraCabecera` para restablecerlo *después* del capítulo especial. Es importante resaltar el *después* pues debe hacerse cuando el capítulo *ya ha terminado* y o bien se ha empezado el siguiente o bien se ha forzado el final de página con un `\newpage`. TeXiS ya hace esto automáticamente justo antes del primer capítulo (en `Tesis.tex`). Sin embargo, si incluyes algún capítulo especial más adelante en el documento, no debes olvidar restaurar la cabecera.

3.2.6. Dividiendo el documento en partes

En ocasiones la estructura del documento tiene dos o más partes claramente diferenciadas. Por ejemplo un libro puede tener una primera parte de conceptos básicos con unos pocos capítulos y otra de conceptos avanzados con el resto.

LaTeX permite especificar distintas partes utilizando el comando `\part`. El resultado es la inserción de una nueva hoja con el número (en romanos) y título de la parte y la adaptación del índice de contenidos para incluir la información de esa nueva parte.

Obviamente, TeXiS también permite la inclusión de distintas partes (y este manual las tiene a modo de ejemplo). Sin embargo, en vez de utilizar directamente el comando de LaTeX, aconsejamos el uso de comandos del propio TeXiS que tienen funcionalidad adicional.

En concreto, los comandos de TeXiS relacionados con las partes del documento (y que describiremos a continuación) permiten añadir una pequeña descripción de la parte que comienza en su hoja de título y una descripción más larga en la parte trasera (sólo si el documento está configurado "a dos caras", especificando `twoside` en el `documentclass` del principio del documento).

TeXiS también se preocupa de que en el índice de contenidos del PDF final la bibliografía (y en caso de existir la última hoja con la frase célebre) *no* aparezcan ligados a la última parte del documento, sino que estén en su mismo nivel.

Dicho todo esto, aconsejamos que, igual que se hace en el código de este manual, existan ficheros para definir cada una de las partes (en el manual se llaman `Capitulos/Parte1.tex`, etc.). Estos ficheros se incluyen desde el documento maestro justo antes del primer capítulo de esa parte.

Los comandos de TeXiS relacionados con las partes del documento son cuatro:

- `\partTitle`: permite especificar el título de la parte que comenzará.
- `\partDesc`: para indicar el texto descriptivo que aparecerá en la "portada" de esa parte. Es opcional; si no se indica, no aparecerá descripción.
- `\partBackText`: sirve para especificar el texto que aparecerá en la parte trasera de la hoja que delimita esa nueva parte. Es responsbilidad del autor asegurarse de que ese texto entra perfectamente en una única cara. Igual que el anterior, es opcional.
- `\makepart`: tras indicar el título y, opcionalmente, descripción y texto trasero, este comando construye la hoja que define esa parte del documento. Si se desea crear una parte sin numerar (lo que en LaTeX suele conseguirse con la versión "con estrella" del comando), se puede utilizar `\makespart` (la `s` solicita la versión *starred*).

A modo de ejemplo este manual contiene tres partes; la primera de ella cubre los tres primeros capítulo y tiene tanto descripción como texto en la parte trasera. La segunda tiene únicamente una descripción y la tercera y última, para los apéndices, no tiene ni descripción ni texto trasero.

El código LaTeX para la definición de la primera parte es:

```
\partTitle{Conceptos básicos}

\partDesc{Esta primera parte del manual presenta los conceptos
  básicos de \texis. Contiene un capítulo de introducción,
  seguido de una descripción de la estructura de \texis\ y
  cómo se genera el documento final, para terminar con un
  capítulo en el que se describe el proceso de edición sugerido
  y los comandos que \texis\ proporciona para facilitar dicho
  proceso.}

\partBackText{En realidad la división por partes del manual no
  aporta demasiado al lector; se ha dividido en varias partes
  debido a que, en la práctica, el código de este manual sirve
  como ejemplo de uso de \texis.

  En un contexto distinto, es posible que un manual de este
  tipo no habría tenido estas partes así de diferenciadas.}

\makepart
```

3.3. Programando en LaTeX

Uno de los aspectos que diferencia a LaTeX de los sistemas ofimáticos tradicionales para creación de documentos es el modelo subyacente que utiliza. En realidad, todo lo que el autor escribe en sus ficheros LaTeX es “*ejecutado*” por el intérprete de LaTeX hasta generar el documento final. Por lo tanto, se puede decir que básicamente, cuando se escribe en LaTeX se “está programando” lo que posteriormente será un programa que generará nuestro documento final. Afortunadamente esa sensación de “programador” no se tiene en condiciones normales durante el proceso de autoría. Sin embargo esta peculiaridad sí se puede aprovechar para facilitar el proceso de edición.

Ya hemos visto en el capítulo anterior un ejemplo de cómo la posibilidad de crear *comandos* de LaTeX nos permite establecer “constantes” que nos evitan tener que escribir palabras que utilizaremos a menudo durante el texto. Sin embargo, profundizando un poco más en el “lenguaje” que hay por debajo (por debajo de LaTeX está TeX) se puede comprobar que pone a nuestra disposición algunas estructuras conocidas por los programadores como los `if`.

3.4. Modos de generación del documento

Aprovechando esto, TeXiS está preparada para admitir dos *configuraciones de generación* o “*compilación*” distintas que, imitando los nombres tradicionales en el desarrollo software, llamamos configuración en modo “release” y en modo “debug” (o de depuración):

- La configuración en modo “release” está pensada para la versión “definitiva”, por lo que genera un fichero con la apariencia final del documento.
- La configuración en modo “debug” puede verse como una versión “borrador”. En este caso el documento incluye ciertos elementos que no se desea incluir en la versión final, como comentarios en el propio texto.

La existencia de estos dos modos de compilación puede sonar extraña al principio. En realidad, su utilidad depende del modo de escribir el documento de cada uno. En nuestro caso, los capítulos de la tesis se escribieron en un proceso “iterativo” de tal forma que incluíamos comentarios que queríamos que aparecieran al imprimir “la versión de depuración”, pero no queríamos preocuparnos de tener que recordar borrar llegado el momento de imprimir la versión final. Por otro lado, cuando el documento es escrito por más de un autor (como este manual), la posibilidad de poner comentarios fácilmente descartables es especialmente útil.

Los ficheros descargados están configurados para compilar la versión definitiva; para cambiarla a la versión de “depuración”, basta con cambiar el fichero `config.tex` del directorio raíz. En cierto momento al principio del fichero aparecen las líneas siguientes.

```
% Comentar la línea si no se compila en modo release.
% TeXiS hará el resto
\def\release{1}
```

Para generar el fichero con la configuración de depuración, basta con comentar la línea en la que se "define" el símbolo `release`[3].

El primer efecto inmediato es que la plantilla añade automáticamente como pie de página el texto:

Borrador – 3 de junio de 2010

De esta forma, si tienes varias versiones imprimidas puedes estar tranquilo de que no se te mezclarán, pues además de marcar que es un borrador, aparece la fecha en la que se generó el fichero.

```
El juego ``Vampire: the Masquerade'', publicado en 1998,
requirió 12 desarrolladores durante 24 meses, casi dos millones
de dólares y unas 366.000 líneas de código.\comp{300.000 para
el juego, y 66.000 de scripts.}
```

Se convierte en:

El juego “Vampire: the Masquerade”, publicado en 1998, requirió 12 desarrolladores durante 24 meses, casi dos millones de dólares y unas 366.000 líneas de código. (**COMENTARIO: 300.000 PARA EL JUEGO, Y 66.000 DE SCRIPTS.**)

3.4.3. Comando `todo`

Este comando permite añadir comentarios para indicar tareas que aún faltan por hacer. Los informáticos solemos marcar esos comentarios en nuestro código fuente utilizando la “palabra” `TODO`[4].

El comando `\todo` encierra el texto entre llaves y lo antecede con la marca “TODO” en negrita, de forma que el código:

```
Existen autores que piensan que enseñar programación orientada
a objetos en el primer curso de programación (CS1) es
beneficioso para los alumnos\todo{Meter referencias...}.
```

se convierte en la versión de depuración en:

Existen autores que piensan que enseñar programación orientada a objetos en el primer curso de programación (CS1) es beneficioso para los alumnos {**TODO TODO TODO:** Meter referencias...}.

Y, al igual que los anteriores, cuando se compila el documento en “modo release”, el comando no tiene ningún efecto.

Es importante destacar que en los dos comandos que van dentro de los párrafos (`\comp` y `\todo`) *no se debe poner ningún espacio antes del comando.* En caso de ponerse el espacio, éste *aparecería* en la versión Release, cuando el comando no tiene ningún efecto:

```
... beneficioso para los
alumnos \todo{Meter
referencias...}.
```

... beneficioso para los alumnos .

Para que cuando se genera el documento en modo depuración quede bien, el propio comando *añade* el espacio de separación entre el texto que le precede y la apertura de la llave.

---

[4] Que en realidad no tiene nada que ver con la palabra española, sino con las inglesas “*to do*”, que puede traducirse aquí a “*por hacer*”.

Ten en cuenta, que al hacer uso de estos comandos para depuración (`\com`, `\comp` o `\todo`) el documento generado contendrá más texto que el final en *release*. Eso significa que el número de páginas variará, y la maquetación general también. Por tanto, *no* debes utilizar el resultado de la generación en depuración para averiguar, por ejemplo, si una figura queda cerca del punto donde es referenciada, o si en una misma página aparecen dos elementos flotantes.

3.5. Acelerando la compilación

Cuando el documento va teniendo más y más páginas, compilarlo una y otra vez hasta dar con el tamaño exacto que queremos darle a una imagen, o para ver si una referencia queda bien generada a partir de la entrada en el `.bib` puede llevar demasiado tiempo.

Para evitarlo, TeXiS permite, de manera fácil, compilar un único capítulo (o apéndice), que normalmente será aquél en el que se esté trabajando.

Para eso, simplemente hay que indicar qué capítulo se quiere compilar en el fichero `config.tex` utilizando el comando `\compilaCapitulo`[5]. Si en vez de ser un capítulo lo que queremos generar es un apéndice el procedimiento es el mismo, pero utilizando el comando `\compilaApendice`. Observa que *no* debe incluirse el nombre del directorio donde aparecen los ficheros (es decir el “`Capitulos`”), pues el propio comando lo hace por nosotros.

Una vez que el capítulo se termina de escribir y se pasa al siguiente, se querrá añadir el `\compilaCapitulo` para el nuevo capítulo (y anular el otro). En nuestro caso, en vez de eliminar el comando del capítulo anterior, lo dejamos comentado por si es necesario en el futuro. Es por ello que al final de la redacción del documento, se tiene una línea por cada uno de los capítulos:

```
% Descomentar la línea para establecer el capítulo que queremos
% compilar

% \compilaCapitulo{01Introduccion}
% \compilaCapitulo{02EstructuraYGeneracion}
% \compilaCapitulo{03Edicion}
% \compilaCapitulo{04Imagenes}
% \compilaCapitulo{05Bibliografia}
% \compilaCapitulo{06Makefile}

% \compilaApendice{01AsiSeHizo}
```

---

[5] El comando sólo puede invocarse una vez, por lo que no es válido si se quiere compilar un grupo determinado de capítulos.

3.6. Editores de LaTeX y compilación

Existen numerosas alternativas para editar los ficheros de LaTeX(ver Flynn, 2005, sec. 2.3), y si has escrito ya algún artículo, posiblemente ya tengas uno “favorito”. Aunque el editor parezca poco importante (al fin y al cabo lo importante es tu documento), en realidad pasarás mucho tiempo utilizándolo, viendo sus colores, pulsando sus botones, y activando sus teclas rápidas.

Evidentemente TeXiS no obliga a utilizar ningún editor en concreto (faltaría más), aunque es posible que necesites hacer algunos cambios en los ficheros para que se adecúen a lo que espera el editor. Esto es especialmente cierto si pretendes generar el documento final utilizando alguna opción del editor.

En la sección 2.6 mostrábamos cómo compilar todos los `.tex` desde la línea de comandos. Sin embargo, reconocemos que esto no es lo más cómodo[6]. Por lo tanto, si el editor que tienes está preparado para LaTeX (no utilizas el Bloc de notas...), es muy posible que tenga algún botón o tecla rápida para compilar el fichero abierto, ya sea con `latex` o `pdflatex`.

Pues bien, en ese caso, debes comprobar cómo funciona exactamente el editor, ya que muy posiblemente, el fichero que estarás editando cuando quieras generar el documento no será el documento *maestro* (es decir, el que en la plantilla hemos llamado `Tesis.tex`, y que contiene el punto de entrada e incluye todos los demás). Por lo tanto, debes mirar de qué manera puedes hacer que el fichero que se envía a `latex` sea el documento maestro. Por ejemplo, WinEdt[7] permite crear “proyectos” donde se añaden ficheros y se especifica cuál es el documento maestro; cuando se pulsa el botón de compilar, independientemente del fichero activo en el editor, se manda compilar el documento maestro.

Como se describe en la sección A.1, nosotros utilizamos emacs (Stallman, 2007) para crear nuestros ficheros LaTeX. Como no podía ser de otro modo, TeXiS está preparado para integrarse con él, en particular con el modo AucTeX que permite una edición cómoda de ficheros TeX (López, 2004). En concreto, este modo dispone de una combinación de teclas para lanzar la generación del documento final. En condiciones normales eso implica enviar al programa `latex` el fichero que se está editando; sin embargo, en nuestro caso lo normal es que el fichero *maestro* que hay que utilizar no es el que se está editando, sino el fichero `Tesis.tex`. Para que funcione como queremos, basta con añadir al final de los ficheros `tex` unas indicaciones para que AucTeX utilice ese fichero como fichero maestro:

```
% Variable local para emacs, para que encuentre el fichero
% maestro de compilación y funcionen mejor algunas teclas
```

---

[6] TeXiS tiene un fichero `Makefile` para la compilación en un único paso, que es explicado en el capítulo 6.

[7] `http://www.winedt.com/`

```
% rápidas de AucTeX
%%%
%%% Local Variables:
%%% mode: latex
%%% TeX-master: "../Tesis.tex"
%%% End:
```

Esta “coletilla” no es necesaria si utilizas cualquier otro editor. Sin embargo TeXIS las tiene añadidas en todos los ficheros (y también en los ficheros de los capítulos y apéndices de este manual). Las líneas anteriores, además, son utilizadas por otras combinaciones de teclas de AucTeX, como las que permiten navegar por todas las secciones del documento.

3.7. Control de versiones

Como veremos en el capítulo 6, el fichero `Makefile` contiene algunos objetivos para realizar copias de seguridad de todos los ficheros del documento.

Sin embargo en el mundo del desarrollo software es habitual utilizar sistemas de control de versiones. Estos sistemas gestionan las distintas versiones por las que van pasando los ficheros durante todo el proceso de desarrollo. La necesidad de estas herramientas está ampliamente conocida, no sólo porque sirven como medio de copia de seguridad que permite *volver hacia atrás* ante algún fallo, sino porque permite el trabajo simultáneo de dos o más personas[8].

Existen varias alternativas para el control de versiones, tanto comerciales como bajo licencia GPL (*General Public License*, Licencia Pública General de GNU) o similares. El sistema por excelencia dentro del software libre fue durante muchos años CVS (*Control Version System*, Sistema de Control de Versiones) (Vesperman, 2003), aunque hoy por hoy ha sido desbancado por Subversion (Collins-Sussman et al., 2004). Entre las herramientas comerciales, destacan SourceSafe de Microsoft[9], Perforce [10] y AccuRev[11].

Aunque es una decisión que los autores del documento tendrán que tomar, aconsejamos el uso de uno de estos sistemas[12]. Una vez que se tiene configurada la máquina servidora que aloja el control de versiones (ver notas bibliográficas), se suben los ficheros *fuente* del documento, que pasarán a estar bajo el control del servidor, lo que permitirá recuperar el estado del documento en cualquier momento pasado (por lo que sirve también como copia de seguridad).

---

[8] Aunque esto en la redacción de una tesis no suele tener sentido, sí puede ser necesario en la elaboración de manuales, cuadernillos de prácticas u otros documentos para los que TeXIS puede utilizarse.

[9] `http://msdn.microsoft.com/ssafe/`

[10] `http://www.perforce.com/`

[11] `http://www.accurev.com/`

[12] En nuestro caso, utilizamos CVS para la escritura de las tesis, mientras que para la elaboración de la plantilla (y manual), utilizamos Subversion.

Un punto importante es hacer que el sistema de control de versiones *ignore* los ficheros que son resultados de la generación del fichero final (el PDF). Cuando se compila el documento, LaTeX genera numerosos ficheros temporales (con extensiones como `.aux` o `.bbl`) que *no* deben subirse al sistema control de versiones. Cuando se utiliza CVS se elimina el problema creando en los directorios un fichero de texto llamado `.cvsignore` que contiene todos los ficheros que deben ser ignorados. A pesar de que en la elaboración de la plantilla no utilizamos CVS, TeXIS incorpora esos ficheros para que puedan utilizarse en el proceso de redacción de los documentos.

Si en vez de utilizar CVS estás utilizando Subversion, puedes hacer que éste ignore los ficheros contenidos en el archivo `.cvsignore` ejecutando la siguiente orden:

```
svn propset svn:ignore -F .cvsignore .
```

en cada uno de los directorios que contengan el fichero. La orden lo que hace es establecer la propiedad (`propset`) concreta para que el Subversion ignore (`svn:ignore`) los ficheros que se indican en el fichero (`-F`) `.cvsignore`.

Notas bibliográficas

La idea de los dos modos de compilación de la Tesis surgió de forma natural dada la experiencia en el proceso de desarrollo en C++, donde los entornos integrados de desarrollo suelen proporcionar al menos esas dos configuraciones posibles. La forma de hacerlo posible vino después de inspeccionar el código LaTeX del libro Bautista et al. (1998). La implementación de los comandos no requiere un conocimiento ni mucho menos extenso de las capacidades de TeX; basta con un poco de intuición al ver un ejemplo de `\if`.

No obstante, el lector interesado en aprender TeX a fondo puede encontrar diversos manuales, como "TeX for the Impatient" (Abrahams, 1990), aunque advertimos que se debe estar *muy* interesado para leerlo, ya que en condiciones normales no se utilizará nada de lo aprendido[13]. También se puede consultar Knuth (1986) o Eijkhout (1991).

Con respecto a la utilización de control de versiones, dentro de las opciones libres es muy utilizado el Subversion, cuyo libro de referencia que ya se ha citado en el texto es Collins-Sussman et al. (2004). Para una descripción sencilla de cómo instalar una máquina servidora puede consultarse Kalderon (2007) y Ziegenhagen (2007). En éste último también aparece una somera descripción de algunos paquetes de LaTeX que pueden utilizarse para incluir información relacionada directamente con las versiones de Subversion. Aunque para más información al respecto recomendamos Scharrer (2007) que dedica toda su atención a la descripción de `svn-multi`, uno de los paquetes con más opciones disponibles para ello.

---

[13] A no ser que se quiera construir un paquete con una funcionalidad muy concreta...

En el próximo capítulo. . .

En este capítulo hemos tratado los aspectos más importantes desde el punto de vista de la edición de un documento realizado con TeXIS, describiendo los comandos LaTeX disponibles.

El próximo capítulo aborda el tratamiento de las imágenes. Como se verá, soportar la generación del documento tanto con `latex` como `pdflatex` dificulta la gestión de imágenes, pues cada uno utiliza un formato de fichero distinto. El capítulo explica las distintas opciones que el usuario de TeXIS tiene para su manejo.

Parte II

Conceptos avanzados

Esta segunda parte del manual contiene capítulos que pueden considerarse “avanzados”, aunque cualquier documento a buen seguro hará uso de los conceptos que en ellos se presentan.

Un primer capítulo explica la gestión de las imágenes que TeXiS espera que se utilice. El manual pasa después a explicar cómo añadir bibliografía y acrónimos. Por último, se describe el fichero `Makefile` proporcionado, que ayuda en algunas de las tareas de generación de documentos.

Capítulo 4

Gestión de las imágenes

El alma nunca piensa sin una imagen mental.

Aristóteles

Resumen: Este capítulo describe todos los aspectos relacionados con las imágenes de los documentos. En particular, describe la estructura de directorios que TeXIS aconseja, así como los aspectos relacionados con la diferencia entre los formatos esperados cuando se genera el documento final con `latex` y `pdflatex`.

4.1. Introducción

En este capítulo tratamos todos los aspectos relacionados con añadir imágenes al documento. Aunque en principio es algo bastante sencillo (desde luego mucho más sencillo que añadir una tabla compleja), existen una serie de cosas a tener en cuenta que merecen un capítulo entero en el manual.

En particular, lo que provoca que las imágenes requieran estas explicaciones detalladas es el hecho de que, como ya dijimos en las secciones 1.2 y 2.6, TeXIS te permite generar el documento utilizando tanto `latex` como `pdflatex`.

Idealmente, el usuario final de LaTeX no debería verse influenciado por la aplicación utilizada para generar sus ficheros. Sin embargo, en cierto modo sí se ve afectado; no por el código en sí contenido en los `.tex` sino por los recursos a los que éstos hacen referencia[1]. En concreto, si se utiliza `latex`, las imágenes referenciadas con el comando `\includegraphics` se asume que tienen formato `.eps`, mientras que en cuanto se utiliza `pdflatex`, se admiten `.pdf`, `.png` y `.jpg` (pero no `.eps`).

---

[1] En ciertas ocasiones también puede verse afectado el código, si se utilizan paquetes que únicamente funcionan con una de ellas.

Por lo tanto, cuando utilizamos un código como el siguiente (similar al que hay en la portada para que aparezca el escudo):

```
\begin{figure}[t]
\begin{center}
\includegraphics[width=0.3\textwidth]%
                {Imagenes/Vectorial/escudoUCM}
\caption{Escudo de la Universidad Complutense}
\end{center}
\end{figure}
```

cuando se genera con `latex`, se buscará el fichero `escudoUCM.eps` en el directorio `Imagenes/Vectorial`, mientras que al generarlo con `pdflatex`, se buscará el fichero en ese mismo directorio, con el mismo nombre, pero con extensión `.pdf`, `.png` o `.jpg`.

Esto provoca que el programa utilizado para generar el documento final sea el que *determina* qué tipo de formato debe usarse para almacenar las imágenes. Existen dos soluciones, que trataremos en las secciones 4.3 y 4.4. Antes de eso, la siguiente sección explica la estructura de directorios que TeXIS espera que se utilice.

4.2. Gestión de imágenes

Los ficheros de imágenes se pueden almacenarse donde el autor del documento desee; al añadir la referencia desde el `.tex`, deberá simplemente indicar la ruta correcta del archivo.

Sin embargo, TeXIS propone una estructura determinada que es la que usa este documento. La estructura está elegida de tal forma que facilita la solución del problema de la generación utilizando tanto `latex` como `pdflatex`, por lo que aunque pueda parecer arbitraria, tiene cierto sentido.

La estructura que proponemos empieza con el directorio `./Imagenes`, donde aparecen los siguientes directorios:

- `./Vectorial`: contiene los ficheros correspondientes a imágenes vectoriales.
- `./Bitmap`: contiene los ficheros correspondientes a imágenes de mapas de bits.
- `./Fuentes`: en este directorio aparecen los "fuentes" de las imágenes. Así, si se crean imágenes con Microsoft Visio, Power Point, o Corel, en este directorio irían los ficheros nativos de esos programas. Estos ficheros *no* serán leídos en el proceso de creación del documento final.

Figura 4.1: Figura utilizada para marcar una imagen por hacer.

Cada uno de los directorios anteriores, a su vez, contiene un directorio por capítulo. De esta forma es fácil encontrar los ficheros si se quieren modificar. En el directorio “raíz” se encuentran las imágenes que no pertenecen a ningún capítulo, como la del escudo de la portada. También pueden aparecer otras imágenes que se utilicen en otras partes del documento que no sean los capítulos. Por ejemplo, TeXiS proporciona una imagen que puede ser de utilidad, y que está colocada en ese directorio raiz por ser independiente del capítulo. Es una figura “dummy” que sirve para marcar el lugar en el que debería aparecer una figura o gráfico que aún está por hacer (figura 4.1).

TeXiS incluye el comando `\figura` para facilitar la inclusión de imágenes. En particular, el comando tiene cuatro parámetros: el nombre del fichero (en el que no hay que indicar el directorio `./Imagenes`), los argumentos pasados al `\includegraphics` que suelen tener información sobre el tamaño deseado, la etiqueta con la que luego podrá referenciarse la ilustración, y por último el título que aparecerá en la parte inferior.

Para incluir la figura 4.1 por lo tanto, el comando es[2]:

```
\figura{Vectorial/Todo}{width=.5\textwidth}{fig:todo}%
       {Figura utilizada para marcar una imagen por hacer.}
```

que gracias al `{fig:todo}`, luego puede citarse en el código LaTeX con:

```
La figura~\ref{fig:todo}
muestra\ldots
```

La figura 4.1 muestra…

La figura se añade automáticamente al índice de figuras que aparece al principio del documento, en el que se indica el número de la figura, el texto inferior y la página en la que aparece. Es posible que el texto sea lo suficientemente largo como para que ocupe más de una línea en la entrada

---

[2] Como se ha mencionado, observa que el primer parámetro donde se indica el nombre del fichero *no incluye* ni el nombre del directorio `Imagenes` ni la extensión del fichero.

Programa	Mapa de bits	Vectoriales
`latex`	`.eps`	`.eps`
`pdflatex`	`.png` | `.jpg`	`.pdf`

Tabla 4.1: Formatos de imágenes para `latex` y `pdflatex`

en el índice. Si se desea utilizar un texto más corto para evitarlo, se puede utilizar el comando `\figuraEx` que recibe un parámetro más con el “título corto” o lo que es lo mismo, con el texto alternativo que aparecerá en el índice.

4.3. Formato de las imágenes

Recuperamos en esta sección el problema anteriormente comentado sobre los formatos de las imágenes. Como ya dijimos en la sección de introducción, el uso de `latex` o `pdflatex` determina los formatos de los ficheros que deben utilizarse para las imágenes, según la tabla 4.1.

Esto significa que, en principio, en el momento de añadir la primera imagen al documento, se debe decidir qué programa se utilizará para generarlo, y utilizar el formato de imagen adecuado a él. Si tienes claro qué programa utilizarás, la solución es así de simple. Almacena las imágenes en el formato adecuado según la tabla 4.1.

Desgraciadamente, lo habitual es no encontrarse en esa situación. Normalmente cuando se comienza a escribir, es muy difícil pronosticar cuál de los dos se utilizará, y por lo tanto no quieres decantarte por ninguno. No estás seguro de cuál quieres, o puede que quieras poder generarlo de las dos formas, debido a alguna restricción del servicio de publicaciones.

Por lo tanto, la mejor solución es, simplemente, permitir ambas alternativas. Para eso lo más fácil es *duplicar* las imágenes, es decir, mantener tanto la copia que será leída por `latex` como la que utilizará `pdflatex`.

Esta duplicación, no obstante, no suele ser aconsejable, pues (además de consumir más espacio) es propensa a errores: si hay que cambiar una imagen, lo habitual será abrir el fichero de `./Imagenes/Fuentes`, y luego “exportarlo” al formato nativo. En ese momento, es fácil olvidar generar *los dos* ficheros.

Por lo tanto, hay dos soluciones rápidas y fáciles:

- Decidir al principio qué programa se utilizará y utilizar siempre los formatos que éste espera, según la tabla 4.1. Tiene la desventaja de que no se podrá (fácilmente) cambiar el programa generador, pues se necesitará crear las imágenes en el formato esperado por la nueva aplicación.
- No atarse al uso de ninguno de los dos, y duplicar los ficheros de for-

ma que todas las imágenes se guardan dos veces, en cada uno de los formatos esperados por ambos programas. Su desventaja es la duplicación de los ficheros, con los problemas de coherencia que eso puede provocar.

Ninguna de las dos alternativas es óptima, por lo que TeXIS proporciona la solución alternativa descrita en la sección siguiente. Si decides utilizar alguna de las opciones fáciles anteriores, puedes omitir la lectura de la misma.

4.4. Imágenes independientes del programa generador

En general conviene evitar duplicar los datos almacenados en disco para evitar problemas de incoherencias. Por eso, cuando no se quiere limitar la generación del documento final a sólo uno de los dos programas, `latex` o `pdflatex`, TeXIS desaconseja almacenar en disco cada imagen en los dos formatos exigidos por ellas.

TeXIS está preparada para que se almacenen en el directorio de las imágenes únicamente las soportadas por `pdflatex` (es decir, ficheros `.pdf` para imágenes vectoriales y `.png` o `.jpg` para mapas de bits). Obviamente, si se hace así, al utilizar `latex` para generar, dará error al no encontrar los ficheros de imágenes correspondientes. Y aquí es donde entra en acción el fichero `Makefile` incluido (que se explica ampliamente en el capítulo 6) cuando se ejecuta con el objetivo `latex`:

```
$ make latex
```

antes de invocar a `latex`, convierte todos los ficheros `.pdf` que hay en el directorio `./Imagenes/Vectorial` a ficheros `.eps`, y todos los `.jpg` y `.png` de `./Imagenes/Bitmap` a `.eps`, para que `latex` los encuentre.

Para realizar la conversión, se utilizan las aplicaciones `pdftops` y `sam2p` que deben estar accesibles en el PATH. Esa es la razón por la que, como mencionábamos en la sección 1.2, TeXIS anima al uso de sistemas Linux: las aplicaciones anteriores están disponibles en este sistema operativo (aunque puede que no se instalen directamente en algunas distribuciones), mientras que en Windows normalmente no están.

4.5. Gestión de imágenes y control de versiones

La solución propuesta en la sección anterior hace que la plantilla espere que dentro del directorio `./Imagenes/Vectorial` aparezcan ficheros `.pdf` y en `./Imagenes/Bitmap` se encuentren ficheros con extensión `.png` o `.jpg`.

En el proceso de generación cuando se utiliza `latex`, se convierten todos esos ficheros a `.eps` para que el programa encuentre las imágenes en el formato que éste espera, por lo que en los directorios anteriores aparecerán ficheros `.eps` *generados automáticamente*.

TeXiS está configurado para que, en caso de utilizar un sistema de control de versiones, éste *ignore* esos ficheros generados (ver una explicación detallada en la sección 3.7). De esta forma, el usuario no es “molestado” en los momentos de las actualizaciones con mensajes indicando que hay nuevos ficheros en el directorio de las imágenes que no han sido subidos al servidor.

Sin embargo, esta característica debe *anularse* si se utiliza una solución distinta a la indicada en el apartado anterior. En particular, se debe *eliminar* el fichero `.cvsignore`[3] si:

- Se decide al principio de la redacción del documento que se va a utilizar `latex` para su generación (y nunca `pdflatex`), y por lo tanto se usarán siempre ficheros con extensión `.eps`.
- Se decide no utilizar la característica de conversión automática de imágenes, y se duplican los ficheros, guardando siempre tanto la copia leída por `latex` como por `pdflatex`.
- Se decide que las imágenes se guardarán siempre en `.eps` y se cambia el `Makefile` para que, en caso de utilizar `pdflatex` las convierta al formato utilizado por éste (consulta la sección 6.3.1 si estás en este caso).

4.6. Imágenes divididas

Somos conscientes de que esta sección incumple lo que comentamos al principio del manual de lo que TeXiS *no era*. Decíamos en la sección 1.3 que esto *no* era un manual de LaTeX, y lo seguimos manteniendo. Sin embargo, en este apartado incumplimos momentáneamente esa promesa para explicar brevemente cómo incluir varias figuras dentro de un entorno flotante (típicamente otra figura).

En este manual ya ha aparecido un ejemplo. La figura 2.1 de la página 15 mostraba en realidad dos capturas distintas, cada una de ellas con un subtítulo distinto.

El código LaTeX de esa figura era:

```
\begin{figure}[t]
  \centering
  %
```

---

[3] O no establecer la propiedad `svn:ignore` con él si se utiliza Subversion.

```
  \subfloat[][Propiedades del documento]{
     \includegraphics[width=0.42\textwidth]%
                     {Imagenes/Bitmap/02/PropiedadesPDF}
     \label{cap2:fig:PropiedadesPDF}
  }
  \qquad
  \subfloat[][Tabla de contenidos]{
     \includegraphics[width=0.42\textwidth]%
                     {Imagenes/Bitmap/02/IndicePDF}
     \label{cap2:fig:TocPDF}
  }
 \caption{Capturas del visor de PDF\label{cap2:fig:pdf}}
\end{figure}
```

La idea general es crear un entorno figura tradicional, pero no poner en ella directamente el `\includegraphics`, sino subdividir ese entorno figura en varias partes. A cada una de ellas se le da una etiqueta diferente para poder referenciarlas, una descripción, etcétera. Un esquema general sería:

```
\begin{figure}[t]
  \centering
  %
  \subfloat[<ParaElIndice1>][<Caption1>]{
     % Contenido para este "subelemento" (podrá ser una
     % figura, una tabla, o cualquier otra cosa).
     \includegraphics[width=5cm]{ficheroSinExtension}
     \label{fig:etiqueta1}
  }
  \subfloat[<ParaElIndice2>][<Caption2>]{
     % Contenido para este "subelemento" (podrá ser una
     % figura, una tabla, o cualquier otra cosa).
     \includegraphics[width=5cm]{ficheroSinExtension}
     \label{fig:etiqueta2}
  }
 \caption{Descripción global para la figura}
 \label{Etiqueta para toda la figura}
\end{figure}
```

El sistema automáticamente decide cuándo poner la siguiente figura al lado, o en otra línea, en base a si entra o no. Es posible forzar a que se ponga en una línea nueva si se deja una línea en blanco en el `.tex`. Esto tiene la repercusión de que no deberías dejar líneas en blanco en ningún momento dentro del entorno flotante, para evitar que se “salte de linea” en las figuras. Si quieres por legibilidad dejar alguna línea, pon un comentario vacio (como hemos hecho con el ejemplo anterior después del `\centering`).

La separación entre dos figuras que se colocan en la misma fila puede ser demasiado pequeña. Para separarlas un poco más, puedes poner `\qquad` entre el cierre llaves de un `\subfloat` y el siguiente. Ese era el cometido del `\qquad` que aparecía en el ejemplo de las figuras visto anteriormente.

Por otro lado, el `\subfloat` tiene dos "parámetros", que se colocan entre corchetes justo después. En realidad ambos son opcionales. Podríamos poner directamente:

```
\subfloat{ <comandos para el subelemento> }
```

pero en ese caso no se etiquetará con una letra.

El texto que se pone entre los primeros corchetes se utiliza para el índice de figuras. En teoría, se mostrará en dicho índice primero la descripción global de la figura, y luego la de cada subelemento. Si no quieres que ocurra, deja en blanco el contenido del primer corchete. En la práctica, el índice de figuras no tiene "niveles", por lo que si se pone la descripción de la subfigura, ésta no aparecerá en el mismo.

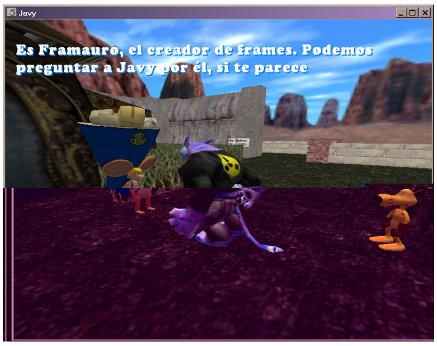
El segundo corchete recibe el texto con la descripción del subelemento, es decir lo que aparecerá junto a la letra identificativa. Si lo dejas vacío (pero poniendo los corchetes), saldrá la letra, sin texto. Si ni siquieras pones los corchetes, no saldrá tampoco la letra.

Por último, debes saber que puedes referenciar de forma independiente cada uno de los subelementos. Para eso, basta con `\ref{etiqueta}`, siendo la etiqueta una definida mediante `\label` *dentro* del `\subfloat`. Se puede entonces referenciar utilizando el comando `\ref` tradicional (que hará que aparezca la letra correspondiente junto con el número de figura), o el comando `\subref`:

```
La figura~\ref{cap2:fig:pdf}
tiene dos partes. La
parte izquierda es la figura~
\ref{cap2:fig:PropiedadesPDF}
y a la derecha está la
\subref{cap2:fig:TocPDF}.
```

La figura 2.1 tiene dos partes. La parte izquierda es la figura 2.1a y a la derecha está la (b).

Como ya hemos dicho, se pueden poner varias "filas" de imágenes en la misma figura. Se puede forzar este comportamiento añadiendo líneas en blanco dentro del entorno `\figure` (lo que será interpretado por LaTeX como un nuevo párrafo), aunque también se añadirán automáticamente si LaTeX detecta que no entra todo en una única línea. La figura 4.2 (extraída de Gómez-Martín, 2008a) es un ejemplo de esto. Como se ve en su código LaTeX que se muestra a continuación, cada una de las imágenes ocupa el 45 % del ancho de la página, por lo que únicamente entran dos figuras por fila. A pesar de que no existe ninguna línea en blanco en el código, las imágenes se colocan en dos filas distintas.



Es Framauro, el creador de frames. Podemos preguntar a Javy por él, si te parece

(a) Estudiante y Javy dirigiéndose al barrio de clases

(b) Estudiante enfrente de Framauro

(c) Estudiante interactuando con la pila de operandos

(d) Estudiante hablando con Javy

Figura 4.2: Ejemplo de uso de `subfloat`.

```
\begin{figure}[t]
  \centering
%
  \begin{SubFloat}
    {\label{fig:cap4:barrioclases}%
     Estudiante y Javy dirigiéndose al barrio de clases}%
    \includegraphics[width=0.45\textwidth]%
                    {Imagenes/Bitmap/04/Javy1BarrioClases}%
  \end{SubFloat}
\qquad
  \begin{SubFloat}
    {\label{fig:cap4:framauro}%
     Estudiante enfrente de Framauro}%
    \includegraphics[width=0.45\textwidth]%
                    {Imagenes/Bitmap/04/Javy1Framauro}%
  \end{SubFloat}
  % La siguiente no entra; ira en otra 'linea'
  \begin{SubFloat}
    {\label{fig:cap4:pilaops}%
     Estudiante interactuando con la pila de operandos}%
    \includegraphics[width=0.45\textwidth]%
                    {Imagenes/Bitmap/04/Javy1PilaOperandos}%
  \end{SubFloat}
\qquad
  \begin{SubFloat}
    {\label{fig:cap4:estudianteyjavy}%
     Estudiante hablando con Javy}%
    \includegraphics[width=0.45\textwidth]%
                    {Imagenes/Bitmap/04/Javy1Javy}%
  \end{SubFloat}
\caption{Ejemplo de uso de \texttt{subfloat}.%
         \label{fig:cap4:javy1}}
\end{figure}
```

Notas bibliográficas

En este capítulo hemos descrito cómo se gestionan las imágenes en TeXiS por lo que no existe ninguna fuente relacionada adicional de consulta.

Conviene, eso sí, indicar que las explicaciones al respecto del entorno `\subfloat` dadas en la sección 4.6 distan mucho de estar completas, aunque es cierto que deberían ser suficientes para la mayoría de los casos. El entorno, no obstante, permite hacer muchas más cosas. Se puede consultar el manual oficial para más información[4].

En el próximo capítulo...

El próximo capítulo pasa a describir algunos aspectos sobre la bibliografía. En concreto, veremos que TeXiS define un estilo de bibliografía propio que, si bien no introduce demasiadas diferencias con respecto al habitual, permite añadir algún campo nuevo a las citas, como por ejemplo la dirección Web y la fecha de la última vez que se visitó (o comprobó su existencia).

---

[4] Disponible en `ftp://tug.ctan.org/pub/tex-archive/macros/latex/contrib/subfig/subfig.pdf`.

Capítulo 5

Bibliografía y acrónimos

> *Como un ganso desplumado y escuálido,*
> *me preguntaba a mí mismo con voz*
> *indecisa*
> *si de todo lo que estaba leyendo*
> *haría el menor uso alguna vez en la vida.*
>
> James Clerk Maxwell, sobre su
> educación en Cambridge

Resumen: Este capítulo aclara algunas cosas sobre la bibliografía utilizada en TeXiS, y sobre la infraestructura para la creación de una lista de acrónimos.

5.1. Bibliografía

Para hacer la bibliografía del documento, TeXiS hace uso, como no podía ser de otra forma, de BibTeX. Esto permite una generación bastante sencilla de la misma, utilizando las entradas `@bibitem` correspondientes. El fichero `makefile` explicado en el capítulo siguiente se encarga de invocar a `bibtex`, la aplicación responsable de la correcta creación de la lista de referencias. Si utilizas cualquier otro sistema para generar el documento final (como por ejemplo las proporcionadas por el editor LaTeX que estés utilizando), deberás encargarte de averiguar cómo debes hacerlo.

La sección siguiente localiza el fichero más importante para la construcción de la bibliografía, e indica dónde cambiar los ficheros `.bib` utilizados.

Por otro lado, para añadir en el texto las referencias, TeXiS (y este manual) hacen uso del formato utilizado por el paquete `natbib` que, como se ha podido comprobar a lo largo de estas páginas, se basa en indicar el nombre del autor y el año de la publicación en el propio texto. La sección 5.1.2 explica brevemente las capacidades del paquete.

TeXiS modifica ligeramente el formato de salida de cada una de las referencias para adecuarlas más a nuestros gustos. También hemos añadido algunas capacidades más que pueden añadirse a las mismas. La sección 5.1.3 las describe.

La parte relativa a la bibliografía termina con una breve sección útil si se quiere cambiar el tipo de bibliografía a utilizar.

5.1.1. Ficheros involucrados

El fichero responsable de la generación de la bibliografía en TeXiS es `Cascaras/bibliografia.tex`. Es el responsable de crear el capítulo sin numerar, poner la cabecera especial para que en la parte superior de todas sus páginas aparezca el texto "Bibliografía", etc. Para eso, al final del fichero aparece la invocación al comando `\makeBib` definido en `TeXiS/TeXiS_bib.tex`.

Ya dijimos en la sección 3.2.3 que para cambiar la cita célebre que aparece en el capítulo sin numerar de la bibliografía debemos editar el fichero `Cascaras/bibliografia.tex`. En ese mismo fichero es donde se configuran los archivos donde se deben buscar los `@bibitem` que se referencian en el texto. Para hacerlo, se utiliza el comando `\setBibFiles`:

```
\setBibFiles{%
nuestros,latex,otros%
}
```

Como se puede ver, en este manual las referencias están organizadas en tres ficheros distintos: referencias a trabajos propios (`nuestros.bib`), referencias relacionadas con LaTeX (`latex.bib`) y referencias varias que no entran en ninguna de las dos anteriores (`otros.bib`).

Esos tres ficheros aparecen en el directorio raíz del manual y pueden/deben ser sustituidos por los utilizados en el nuevo documento.

5.1.2. Referencias con `natbib`

Aunque en este manual no se haya hecho un uso demasiado extenso de la bibliografía, es cierto que `natbib` proporciona opciones muy interesantes para utilizarse en textos donde las referencias tengan un peso importante (o al menos más importante que en este manual).

Cuando se utiliza `natbib`[1] en vez de hacer uso de `\cite`, se pueden utilizar otras dos versiones distintas del comando, en concreto `\citet` y

---

[1] El paquete es incluido por TeXiS en `TeXiS/TeXiS_pream.tex`.

Comando	Resultado
`\citet{key}`	Jones et al. (1990)
`\citet*{key}`	Jones, Baker y Smith (1990)
`\citep{key}`	(Jones et al., 1990)
`\citep*{key}`	(Jones, Baker y Smith, 1990)
`\citep[cap. 2]{key}`	(Jones et al., 1990, cap. 2)
`\citep[e.g.][]{key}`	(e.g. Jones et al., 1990)
`\citep[e.g.][p. 32]{key}`	(e.g. Jones et al., p. 32)
`\citeauthor{key}`	Jones et al.
`\citeauthor*{key}`	Jones, Baker y Smith
`\citeyear{key}`	1990

Tabla 5.1: Distintas opciones de referencias con `natbib`

`\citep`. El primero está pensado para hacer referencias en el propio texto y el último para que las referencias aparezcan entre paréntesis.

En vez de hacer una descripción de cada una de las variaciones, la tabla 5.1 contiene las distintas posibilidades. Las opciones están cogidas directamente de la documentación del paquete y asume que existe un `@bibitem` con nombre `key` que describe una referencia escrita por los autores Jones, Baker y Smith en 1990. Notar que el resultado contiene la conjunción española "y" en vez de la inglesa "and" para separar el último autor.

5.1.3. Modificaciones en los `@bibitem`

En TeXiS hemos definido nuestro propio estilo de bibliografía, es decir, el formato de salida de las referencias en el listado que aparece al final del documento. No es demasiado distinto del estilo utilizado por defecto, pero sí tiene ligeras modificaciones.

Las modificaciones más obvias es que se utiliza la versión española del formato, para que aparezca "editor", "páginas" o "Informe Técnico" entre otros.

También hemos añadido "constantes" (o, en nomenclatura de BibTeX, "macros") para una editorial y dos series muy utilizadas en nuestra área, "Springer-Verlag" (`SV`), "Lecture Notes in Computer Science" (`LNCS`) y "Lecture Notes in Artificial Intelligence (subserie de LNCS)" (`LNAI`). De esta forma, una entrada de la bibliografía puede ser:

```
@inproceedings{Ejemplo,
    author = { ... },
   title = { ... },
   ...
   publisher = SV,
  series = LNCS,
  ...
  year = 2009,
  month = jan
}
```

Como se puede ver se han hecho uso de las macros para indicar la editorial y la serie. También se ha utilizado la macro para indicar el mes (`jan` equivale a enero), de forma que la entrada sea independiente del lenguaje utilizado posteriormente en la referencia.

Mucho más interesante es la creación de *dos nuevos campos* que pueden añadirse en las entradas de BibTeX, y que permiten indicar la página Web en la que se puede encontrar la referencia.

En concreto, admitimos un nuevo campo, `webpage` que permite indicar una página Web, y un campo `lastaccess` permite indicar la fecha del último acceso. De esta forma, la referencia, además de los campos e información habituales, incluye la localización.

Por ejemplo, el libro de Subversion citado en un capítulo anterior (Collins-Sussman et al., 2004) está disponible en la Web; para que la URL o dirección Web aparezca en la referencia, se pueden utilizar los nuevos campos:

```
@Book{Subversion,
  author      = {Ben Collins-Sussman and
                 Brian W. Fitzpatrick and
                 C. Michael Pilato},
  title       = {Version Control with Subversion},
  publisher   = {O'Reilly},
  year        = 2004,
  isbn        = {0-596-00448-6},
  webpage     = {http://svnbook.red-bean.com/},
  lastaccess  = {Octubre, 2009}
}
```

El estilo añade la cadena "Disponible en" antes de la url (que se añade con el comando `\url` para su correcta división en líneas) y entre paréntesis incluye la fecha del último acceso. El aspecto final de la referencia del libro anterior es:

Collins-Sussman, B., Fitzpatrick, B. W. y Pilato, C. M. *Version Control with Subversion*. O'Reilly, 2004. ISBN 0-596-00448-6. Disponible en `http://svnbook.red-bean.com/` (último acceso, Octubre, 2009).

También hemos añadido un nuevo tipo de entrada para referenciar artículos de la Wikipedia. Puede ser discutible si es adecuado o no utilizar citas de la Wikipedia en documentos académicos como tesis o trabajos de fin de master, pero si eres de los que opinan que debe citarse todo lo que uno utilice y utilizas la Wikipedia, puede que quieras usar esta nueva entrada que hay en TeXiS. Por ejemplo, la entrada de LaTeX de la Wikipedia (Wikipedia, LaTeX) se define con:

```
@Wikipedia{LaTeXWikipedia,
   author     = {Wikipedia},
   title      = {{LaTeX}},
   wpentry    = {LaTeX},
   language   = {es},
   webpage    = {http://es.wikipedia.org/wiki/LaTeX},
   lastaccess = {Mayo, 2009},
}
```

En el texto la referencia no aparece con el año sino que hace alusión a que es una entrada de la Wikipedia, como se ha podido ver en el parrafo anterior. El resultado final en la lista de referencias es:

Wikipedia (LaTeX). Entrada: "LaTeX". Disponible en `http://es.wikipedia.org/wiki/LaTeX` (último acceso, Mayo, 2009).

5.1.4. Cambio del estilo de la bibliografía

El estilo de las referencias de TeXiS está definido en `TeXiS/TeXiS.bst`, que se incluye en la parte final del fichero `TeXiS_bib.tex`:

```
...
\bibliographystyle{TeXiS/TeXiS}
...
```

Si quieres utilizar un estilo distinto de los disponibles en LaTeX basta con que cambies esa línea para definir ese otro estilo. Por ejemplo:

```
...
\bibliographystyle{abbrv}
...
```

configura la bibliografía para que queden numeradas y se referencien desde el texto con el número entre paréntesis[2]. Ten en cuenta, no obstante, que en ese caso el resultado de los comandos `\citep` y `citet` es el mismo, por lo que puede que necesites reescribir algunas frsaes; además, `\citeyear` y `\citeauthor` dejarán de funcionar. Por último, cambiar el estilo a un estilo predeterminado como ese elimina la posibilidad de incluir referencias a páginas Web y a la Wikipedia explicadas antes.

---

[2] Si quieres que queden entre corchetes ('[', ']'), debes quitar el `[round]` que aparece en la inclusión del paquete `natbib` en el fichero `TeXiS/TeXiS_pream.tex`.

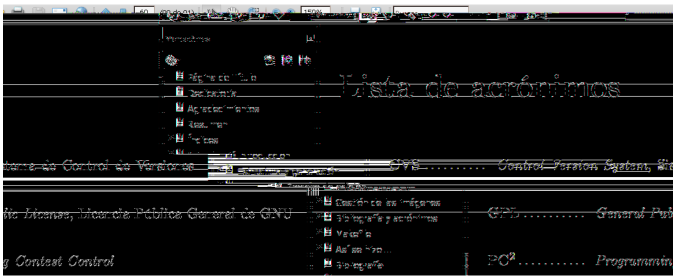


Figura 5.1: Resultado de la lista de acrónimos

Otra posibilidad es que desees ajustar el funcionamiento del estilo proporcionado por TeXiS. En ese caso, puedes editar el fichero del estilo (el antes nombrado `TeXiS/TeXiS.bst`), respetando la sintáxis que espera `bibtex`.

5.2. Acrónimos

Los acrónimos son las *siglas* utilizadas a lo largo del documento. LaTeX permite facilitar su gestión, de manera que se controle automáticamente el momento de la primera aparición de un acrónimo para poner su significado, o para que se genere automáticamente una lista de acrónimos a modo de resumen para ser añadida al final del documento.

Dado que el uso de acrónimos no es tan conocido, de nuevo romperemos nuestra promesa de no explicar aquí aspectos concretos de LaTeX, y describiremos en la siguiente sección el funcionamiento del paquete GlossTeX, que se encarga de la gestión de acrónimos. Después, nos centraremos en el modo en el que TeXiS integra su uso.

5.2.1. Acrónimos con glosstex

Al igual que ocurre con la bibliografía, el uso de acrónimos en el documento supone dos cosas:

- A lo largo del texto habrá que indicar en qué punto se usa un acrónimo, al igual que se indica cuando se utiliza una referencia bibliográfica.
- Al final del texto, tendrá que aparecer una lista con todos los acrónimos utilizados (figura 5.1), al igual que ocurre con las citas.

Dada esta similitud con la bibliografía, no sorprende que para que sea LaTeXquién nos gestione nuestros acrónimos tendremos que hacer uso de una “base de datos” de acrónimos, generalmente en ficheros con extensión `.gdf` (*Glossary Data File*), que son conceptualmente similares a los ficheros `.bib` usados por BibTeX. Como ejemplo, a continuación se indica el contenido (parcial) del fichero usado en este manual:

```
@entry{CVS, , \emph{Control Version System}, Sistema de Control
de Versiones}

@entry{GPL, , \emph{General Public License}, Licencia Pública
General de GNU}
```

Cada entrada se coloca dentro de un “comando” `@entry`, que tiene tres parámetros, separados por comas:

1. *Acrónimo*: en el ejemplo, CVS y GPL. El acrónimo hace también las veces de etiqueta identificativa, para referenciarlo dentro del texto.
2. *Representación corta*: no aparece en ninguno de los dos ejemplos anteriores (se ha dejado en blanco). Es necesaria únicamente si el acrónimo *tiene formato*. Veremos un ejemplo en un instante.
3. *Versión larga*: contiene la descripción completa del acrónimo. Aunque estemos utilizando comas como separadores de cada uno de los tres elementos, al ser éste el último campo, podría contener comas tal y como muestran los ejemplos.

Una vez que se ha poblado el fichero con los acrónimos, a lo largo del texto es posible añadir comandos de GlossTeX para referirse a ellos, algo conceptualmente similar al uso de `\cite` respecto a la bibliografía. A continuación se muestran los comandos disponibles[3], y su resultado en el documento final:

- `\acs{CVS}` (*short*): CVS
- `\acl{CVS}` (*long*): *Control Version System*, Sistema de Control de Versiones
- `\acf{CVS}` (*full*): CVS (*Control Version System*, Sistema de Control de Versiones)
- `\ac{CVS}`: es la más interesante de todas. Funciona como `\acf` la primera vez que aparece en el texto, y como `\acs` el resto de las veces.

Las tres primeras muestran diferentes partes de la entrada definida en el fichero `.gdf`. La más completa es la mostrada por `\acf`, que combina la versión corta con la larga, que coloca entre paréntesis. No obstante, el comando más interesantes es `\ac`, que "recuerda" si ya se introdujo un acrónimo anteriormente en el documento, mostrando su descripción completa únicamente la primera vez[4]. El modo en el que se construye esta descripción (con la versión larga entre paréntesis) es personalizable, si bien el modo de hacerlo queda fuera del alcance de este documento.

---

[3] Para poder usarlos, será necesario haber incluído el paquete `glosstex` en el preámbulo del documento.

Por tanto, el modo de uso recomendado de los acrónimos es añadir en el fichero `.gdf` todos los acrónimos usados en el documento, y luego hacer *siempre* referencia a ellos a través del comando `\ac`. LaTeX incluirá la descripción completa la primera vez, y dejará únicamente el acrónimo todas las demás.

Hemos visto que en los ejemplos descritos, el propio acrónimo *hace las veces de identificador*, de modo que para referenciarlo basta con un mero `\ac{CVS}` o similar. Esto será suficiente la mayor parte de las veces, pero en ocasiones tendremos acrónimos más complejos, como por ejemplo PC^2 (*Programming Contest Control*). El problema, es que para que se muestre correctamente, el código LaTeX del acrónimo es en realidad `PC$^2$`, por lo que tendríamos que referirnos a él como `\ac{PC$^2$}`. Esto no sólo resulta incómodo, sino que además genera un error de compilación, dado que no podemos añadir comandos LaTeX dentro de un identificador.

La solución es hacer uso del segundo campo que dejábamos vacío en las entradas del fichero `.gdf` de ejemplo:

```
@entry{PC2, PC$^2$, \emph{Programming Contest Control}}
```

Cuando se indica dicho campo, GlossTeX hará uso de él para mostrar el acrónimo, y utilizará el primero sólo como identificador. Ahora, para referirnos a él bastará con un `\ac{PC2}` que, la primera vez que se usa, se convierte en un PC^2 (*Programming Contest Control*), y las siguientes en un más corto PC^2.

Aparte de la ayuda proporcionada por GlossTeX para evitarnos tener que escribir la versión completa de nuestros acrónimos, también nos sirve para añadir en la parte final un listado de acrónimos donde se muestra el acrónimo y su descripción larga (tal y como mostraba la figura 5.1). Al igual que ocurre con la bibliografía, sólo aparecerán en la lista aquellos acrónimos que realmente se hayan usado a lo largo del documento. El comando para conseguirlo es:

```
\printglosstex(acr)
```

No obstante, y al igual que ocurre con la bibliografía, para que toda esta infraestructura funcione es necesario ejecutar programas externos (aparte del propio `latex` o `pdflatex`). El proceso completo es el siguiente:

---

[4] El uso manual de `acf` *no* hace que GlossTeX considere que el acrónimo ya se ha "presentado", por lo que si se utiliza primero `acf` y más adelante `ac`, aparecerá de nuevo la versión completa.

1. El uso de comandos `\ac?` genera la inclusión de anotaciones dentro de los ficheros `.aux` generados al compilar los `.tex`.
2. Al igual que se hace con la aplicación `bibtex` para la bibliografía, en este caso usamos el programa `glosstex`[5] que los recorre y genera un nuevo fichero con extensión `.gxs` con información sobre las entradas referenciadas, y un `.gxg` con información de registro (*log*).
3. El fichero `.gxs` debe volverse a procesar usando otro programa, en este caso `makeindex`, que ordena alfabéticamente las entradas utilizadas, y les aplica un formato concreto. Este proceso genera un nuevo fichero, con extensión `.glx`, que, finalmente, contiene todo correcto.
4. El fichero anterior es utilizado por el comando LaTeX `printglosstex` que debe ser incluído dentro de algún fichero `.tex` del documento, y que es conceptualmente similar al `\bibliography{ficheros}` usado para BibTeX.

Todo esto complica el proceso de construcción del documento, que ahora requiere pasos adicionales[6]:

```
$ pdflatex Tesis
$ bibtex Tesis
$ glosstex Tesis acronimos.gdf
$ makeindex Tesis.gxs -o Tesis.glx -s glosstex.ist
$ pdflatex Tesis
```

En la ejecución de `glosstex` es necesario proporcionarle el nombre del fichero (o ficheros) con la "base de datos" de acrónimos. En el caso de `bibtex` esto no es necesario, porque ya se lo proporcionamos directamente en los `.tex` con el comando `\bibliography{ficheros}`. Además, en la ejecución a `makeindex` proporcionamos el parámetro `glosstex.ist` que indica el formato que queremos aplicar a nuestra lista de acrónimos; ten en cuenta que `makeindex` se utiliza para la generación de otros índices, por lo que para mantener la generalidad mantiene fuera dicho formato para que pueda ser adaptado a cada caso.

Para poder hacer uso, por tanto, de los acrónimos, *es necesario* disponer no sólo del paquete LaTeX `glosstex`, sino también de las *aplicaciones* `glosstex` y `makeindex`. En la sección 2.6 veíamos un modo simplificado de compilar el presente documento que no incluía las órdenes para procesar los acrónimos. Si no se ejecutan estas aplicaciones, el documento se generará correctamente, salvo por la lista de acrónimos que aparecerá vacía.

---

[5] Este programa deberá estar instalado en el sistema en el que se esté generando el documento.

[6] Como siempre, también es válido el uso de `latex` en lugar de `pdflatex`, pero el fichero generado (`.dvi`) deberá después ser convertido a PDF.

5.2.2. Acrónimos en TeXiS

Afortunadamente, al utilizar TeXiS prácticamente de lo único que hay que preocuparse es de crear la "base de datos" de acrónimos y de hacer uso del comando `\ac` cuando corresponda. El resto de tareas son gestionadas automáticamente[7].

Para dar soporte al uso de acrónimos, TeXiS se apoya en varios ficheros:

- `TeXiS/TeXiS_acron.tex`: contiene el comando de GlossTeX que añade al documento un nuevo capítulo sin numeración con la lista de acrónimos. De manera predefinida, este capítulo se añadirá al documento *unicamente en modo "release"* (consulta la sección 3.4 para más información). En modo borrador (*debug*) los acrónimos no se incluyen en el documento, de modo que se ahorra algo de tiempo.

- `acronimos.gdf`: éste es el fichero donde se recomienda añadir los acrónimos que se usen. Hace las veces de "base de datos" de acrónimos. Siendo realistas, los ficheros `.tex` de TeXiS *no* dependen de que se utilice este fichero en concreto. Tal y como se explicó en la sección anterior, es la ejecución externa de las herramientas de GlossTeX la que recibirá el nombre en concreto. Sin embargo, TeXiS proporciona un modo automático de construcción, descrito en el capítulo 6, que sí asume este nombre de fichero. La construcción se apoya en la herramienta `make`, de ahí que al principio del documento mencionáramos que resultaba más cómodo hacer uso de plataformas GNU/Linux (que disponen de él) en lugar de Windows.

En la versión en borrador, además de no generarse la lista de acrónimos como un capítulo más, *tampoco* se añaden las descripciones largas de los acrónimos. Es decir, si se utiliza en algún lugar del documento las órdenes de GlossTeX mencionadas previamente (`\ac`, `\acl`, etcétera), éstas se "puentearán" de modo que en el documento final tan sólo aparecerá la propia etiqueta. Por ejemplo, ante `\acl{CVS}`, en lugar de aparecer "*Control Version System*, Sistema de Control de Versiones" se mostrará únicamente "CVS".

Dado que no todos los documentos harán uso de acrónimos, y que, después de todo, generarlos supone un esfuerzo en la fase de compilación no despreciable[8], es posible que en ocasiones no se desee que se incluyan los acrónimos tampoco en la versión final (modo *release*). En ese caso, basta con que se modifique el fichero `config.tex` (el mismo en el que se escogía qué versión se quería compilar) y comentar la linea siguiente:

---

[7] Salvo, naturalmente, la instalación de las propias aplicaciones `glosstex` y `makeindex` que deberá haber realizado el usuario.

[8] Este esfuerzo es real tan sólo si no se hace uso del `Makefile` proporcionado por TeXiS.

```
\def\acronimosEnRelease{1}
```

Si se comenta, TeXiS asumirá que no se desean acrónimos tampoco en la versión final, y no se incluirá el capítulo sin numeración. De nuevo, ten en cuenta que en ese caso los comandos de GlossTeX se puentearán también[9].

En el improbable caso en el que se quiera hacer uso de los acrónimos para que sea el propio sistema el que se encargue de escribirnos el significado la primera vez que se usan, pero no se quiere que aparezca el listado final de los acrónimos, entonces será necesario modificar directamente el fichero `Tesis.tex`, y dejar de incluir `TeXiS/TeXiS_acron`, cuya inclusión es ahora mismo condicional en función de si se usan o no los acrónimos.

Por último, hay que tener en cuenta que cuando se modifica el modelo de compilación (por ejemplo, indicando versión final o borrador, o pidiendo que se añadan o quiten los acrónimos en *release*) es necesario *borrar los ficheros intermedios* generados durante la compilación[10]. Si se está generando el documento haciendo uso de la infraestructura proporcionada por TeXiS a través de su `Makefile` (descrita en el capítulo siguiente), bastará con un sencillo:

```
make clean
```

5.2.3. Más allá de TeXiS

El nombre GlossTeX proviene en realidad de *glossary*, por lo que su objetivo inicial era realizar *glosarios*, no meras listas de acrónimos. Para ser honestos, en las entradas de los ficheros `.gdf` podemos añadir una *descripción completa* con una descripción de la entrada:

```
@entry{PC2, PC$^2$, \emph{Programming Context Control}}
Software de gestión utilizado en los concursos de programación
impulsados por ACM, a través del cual los concursantes envían
sus soluciones, y los jueces acceden a ellas y las valoran.
```

Esa descripción *está fuera* de la entrada `@entry`, y resulta útil en el caso de que quisiéramos mantener un glosario de palabras. TeXiS *no* proporciona soporte para esto, por lo que si lo quieres utilizar, tendrás que añadir la infraestructura necesaria por tu cuenta.

---

[9] Aunque en este caso no debería ser un problema porque si no se quieren los acrónimos en la versión final será porque no se han usado.

[10] Las razones que ocasionan esto quedan explicadas en la sección 6.3.2.

Notas bibliográficas

Existen numerosas publicaciones relacionadas con BibTeX; para una descripción de los tipos de entradas que se soportan, etc., se puede consultar Patashnik (1988a) o Mittelbach et al. (2004). En esta última referencia también puede encontrarse información sobre las posibilidades del paquete `natbib`.

Por otro lado, es también fácil encontrar información sobre cómo cambiar el estilo utilizado (es decir, lo que hemos hecho en TeXiS para añadir el `@lastaccess` o ajustar las cadenas que aparecen). Por ejemplo, una descripción en español es López (2006). También se puede consultar Mittelbach et al. (2004) o Patashnik (1988b).

Respecto a GlossTeX, la fuente principal de información es la disponible en el catálogo de paquetes de TeX. Puedes encontrarla en `http://www.ctan.org/tex-archive/help/Catalogue/entries/glosstex.html`.

En el próximo capítulo...

Con este capítulo terminan los capítulos más importantes del manual, donde se ha contado lo que se debe saber para utilizar TeXiS. El próximo capítulo describe el fichero `Makefile` proporcionado. El fichero permite generar de forma fácil el documento final, utilizando la utilidad `make` disponible en virutalmente todas las plataformas. Somos conscientes de que no todo el mundo querrá utilizar este mecanismo para generar el documento final (muchos usuarios preferirán utililizar las opciones del editor que utilicen); por eso lo hemos puesto al final.

Capítulo 6

Makefile

> *A fuerza de construir bien, se llega a buen arquitecto.*
>
> Aristóteles

Resumen: Este capítulo describe la infraestructura de creación del documento final, apoyada en la herramienta `make` de GNU.

6.1. Introducción

Ya se ha esbozado a lo largo de este manual que la generación del documento final requiere varias etapas, algo que es de hecho inherente al propio LaTeX (o PdfLaTeX). En la sección 2.6 vimos que era necesario la invocación a `pdflatex` tres veces, junto con el uso de `bibtex`. En la sección 5.2.1 se añadió la necesidad de invocar a `glosstex` y a `makeindex` para añadir el listado de acrónimos. El resultado es una generación bastante laboriosa que requiere dar varios pasos en un orden concreto.

El mundo del desarrollo del software ha lidiado con un problema similar (más complejo, de hecho) desde hace décadas, y que se ha ido resolviendo con diferentes herramientas, siendo `make` una de las más extendidas. TeXiS proporciona un fichero `Makefile` para ser utilizado con ella, y simplificar la generación del documento, así como otras labores rutinarias. Dado que la infraestructura `make` no está disponible nativamente en plataformas Windows, sólo podrá utilizarse sobre GNU/Linux y otras variantes de Unix[1]. Incluso aunque se utilizara `nmake`, una herramienta similar a `make` proporcionada con Visual Studio, el `Makefile` de TeXiS no funcionaría dado que hace uso de comandos que sólo están disponibles en Unix/Linux. Es por ello que en el primer capítulo recomendábamos el uso de dicho sistema.

---

[1] Windows dispone de Cygwin que proporciona muchas de las herramientas habituales en Unix. *No* hemos probado TeXiS sobre él; si lo haces, ¡no dudes en contarnos la experiencia!

En la sección siguiente, se describen los diferentes *objetivos* que este `Makefile` proporciona. La sección 6.3 describe brevemente el funcionamiento de algunas de sus partes.

6.2. Objetivos del `Makefile`

En la terminología de `make`, un objetivo es un *grupo de tareas* que se ejecutan en conjunto. En el entorno del desarrollo del software, esas tareas suelen estar enfocadas a la compilación del proyecto, aunque también se incluyen objetivos para, por ejemplo, borrar los ficheros intermedios, o instalar el programa recién compilado.

El `Makefile` de TeXiS sigue esa misma filosofía, proporcionando los siguientes objetivos:

- `pdflatex`: es el objetivo por defecto. Genera el documento utilizando PdfLaTeX. Se encarga de generar la bibliografía, los acrónimos, y de compilar varias veces el documento para que se actualicen correctamente las referencias.
- `latex`: genera el documento utilizando LaTeX, y convierte el `.dvi` resultante en `.pdf`. Tiene en cuenta las necesidades en cuanto al formato de las imágenes descritas en la sección 4.4, por lo que se convierten automáticamente a formato `.eps`.
- `imagenes`: se encarga de convertir las imágenes (tanto vectoriales como de mapas de bits) a formato `.eps`. Normalmente este objetivo no necesitará ser lanzado manualmente nunca; el objetivo `latex` anterior lo hace por nosotros.
- `imagenesvectoriales`, `imagenesbitmap`: convierte o bien las imágenes vectoriales, o bien las de mapas de bits a formato `.eps`. Igual que antes, normalmente no se necesitan invocar manualmente.
- `fast`: genera el documento de manera rápida utilizando PdfLaTeX. Se limita a generarlo una única vez, sin invocar a BibTeX ni a GlossTeX. Está pensada para la compilación "del día a día" cuando se añade algo de texto y se quiere ver rápidamente el resultado, sin preocuparnos de que las referencias queden correctamente actualizadas. Si este objetivo se combina con el comando `\compilaCapitulo` descrito en la sección 3.5, la compilación puede resultar muy rápida.
- `fastlatex`: similar al anterior, pero realiza la generación utilizando LaTeX. En este caso, el `.dvi` sigue convirtiéndose a `.pdf`.

- `clean`: elimina todos los ficheros intermedios creados durante la generación del documento. Este objetivo resulta interesante cuando se quiere construir el documento completamente, sin basarse en información previa. Es, de hecho, necesario lanzarlo cuando se compila el documento con la lista de acrónimos por primera vez (o cuando deja de hacerse). Si no se ha modificado la definición de la constante `\acronimosEnRelease` (consulta la sección 5.2.2 para información sobre ella), esto ocurrirá siempre que se cambie el modo de configuración de borrador a versión final (sección 3.4). Ten en cuenta que este objetivo *borra también* los ficheros `.eps`, por lo que si has modificado el modelo de gestión de imágenes (para que los originales sean `.eps` que se conviertan a `.pdf` si se usa PdfLaTeX, consulta la sección 6.3.1 para más información) entonces tendrás que modificar también este objetivo.
- `distclean`: similar a la anterior, pero también borra el fichero `.pdf` generado, y los ficheros de copia de seguridad de los `.tex` creados por los editores de texto más habituales (con extensiones `.tex~` y `.backup`).
- `crearZip`: genera el documento (con PdfLaTeX) y crea un fichero `.zip` con él y todos los fuentes (incluído imágenes). El fichero es útil para distribuirlo a revisores que tengan intención de cambiar y regenerar el documento.
- `crearVersion`: similar al anterior, pero copia el .zip en el subdirectorio `VersionesPrevias` incluyendo en el nombre la fecha y hora actuales. Es útil para realizar copias de seguridad locales o para “congelar versiones” en “hitos” concretos de la escritura.
- `crearBackup`: genera el documento, y hace una copia de *todo* el directorio (incluyendo el subdirectorio `VersionesPrevias` mencionado antes) comprimiéndola en un fichero `.zip` que copia en el *directorio padre* del actual. Está pensado para hacer una copia de seguridad completa que luego sea guardada en algún otro lugar.
- `ayuda` o `help`: muestra una descripción de todos los objetivos anteriores.

En el día a día, los más útiles son `pdflatex`, `fast` y `clean`. Para invocar a cualquiera de ellos en un entorno GNU/Linux donde `make` esté disponible bastará con:

```
$ make <objetivo>
```

En el caso de que se quiera realizar la generación usando PdfLaTeX, al ser el objetivo por defecto (el primero que aparece en el `Makefile`) no es necesario especificar nada:

```
$ make
pdflatex Tesis
This is pdfTeXk, Version 3.141592-1.40.3 (Web2C 7.5.6)
[...]
```

Si decides que tu herramienta de generación por defecto sea LaTeX, seguramente quieras colocar el objetivo `latex` delante para convertirlo en el que se ejecute por defecto.

6.3. Funcionamiento interno

En principio, el fichero `Makefile` debería funcionar por sí solo si se siguen los convenios de TeXiS descritos en los capítulos previos, por lo que la sección anterior debería ser suficiente para un usuario normal. Aquí describiremos algunos detalles internos que pueden ser útiles en algunos (idealmente pocos) casos.

6.3.1. La compilación de las imágenes

En el capítulo 4 se mencionaba que TeXiS es capaz de independizarse de la diferencia entre los formatos de imágenes aceptados por PdfLaTeX y LaTeX siempre que el autor siga un determinado convenio en el modo de gestionarlas y utilice la infraestructura de generación, es decir el fichero `Makefile` al que se refiere este capítulo.

Dado que nosotros hacemos normalmente uso de PdfLaTeX, TeXiS asume que de manera nativa se querrá utilizar éste en lugar de LaTeX, por lo que muestra una clara tendencia hacia el formato de imágenes que `pdflatex` soporta de manera nativa.

En concreto, el `Makefile` asume que las imágenes se proporcionan en formato `.pdf` para el caso de las vectoriales, y en formato `.jpg` o `.png` para los mapas de bits. Si las imágenes se desarrollaron originalmente con Kivio, Corel, Visio, Gimp o Photoshop, es responsabilidad del propio autor convertirlas a los formatos soportados por PdfLaTeX. De esa manera, el `Makefile` *no* necesitará realizar ningún tipo de transformación de imágenes en el objetivo por defecto `pdflatex`.

En el caso de que se desee utilizar LaTeX, la situación es más complicada. El `Makefile` tendrá que buscar las imágenes (tanto vectoriales como de mapas de bits) y convertirlas a `.eps`. De eso se encargan los objetivos `imagenesbitmap` y `imagenesvectoriales` respectivamente.

Ambos se apoyan en el convenio de directorios propuesto por TeXiS, en el que se asume que las imágenes vectoriales estarán en el directorio `Imagenes/Vectorial`, y las de mapas de bits en `Imagenes/Bitmap`, con un subdirectorio dentro por capítulo[2]. El `Makefile` encadena una serie de invocaciones a otros `Makefile` y algunas llamadas a *scripts* del *shell* (`bash`) para realizar la conversión. En última instancia, el responsable de dicha conversión es un *script* llamado `update-eps.sh`, del que, en realidad, existen dos versiones, uno dentro de `Imagenes/Vectorial` y otro en `Imagenes/Bitmap`, específicos para cada uno de los dos tipos de imágenes. Para convertir los `.pdf` vectoriales a `.eps`, se hace uso de la aplicación `pdftops`, que deberá estar instalada. Por su parte, para convertir las imágenes de mapas de bits `.jpg` o `.png` se usa `sam2p`. Los scripts terminan con error (deteniendo por tanto la generación del documento) si estas herramientas no están disponibles. Se han desarrollado de tal manera que el error únicamente se lance si realmente se necesita convertir alguna imagen. Además, se evita regenerar los ficheros `.eps` si los originales (`.pdf`, `.jpg` o `.png`) no han sufrido cambios desde la última generación. Como ya se dijo previamente, se debe tener en cuenta que el `Makefile` considera a esos ficheros `.eps` como *ficheros generados*, por lo que son eliminados por el objetivo `clean`, y se anima a que se configure el control de versiones (CVS, SVN, etcétera) para que *los ignore*.

Si por alguna razón se prefiere LaTeX a PdfLaTeX, entonces resultará más cómodo utilizar el formato `.eps` como predefinido, de manera que a partir de las imágenes originales (creadas con cualquier programa de dibujo) se generen los `.eps` en lugar de los `.pdf` mencionados antes. Si, además, se quiere mantener la posibilidad de generar el documento con `pdflatex` (aunque sea a costa de trabajar más convirtiendo las imágenes), deberá adaptarse la infraestructura de generación en varios puntos:

- Modificar los ficheros `update-eps.sh` que se encuentran en los directorios `Imagenes/Bitmap` y `Imagenes/Vectorial` para que conviertan los `.eps` “fuente” en `.pdf`. En este caso no merece la pena convertir a `.jpg` o `.png` las imágenes vectoriales; resulta más cómodo convertir todo a `.pdf`. Para eso, se puede utilizar `epstopdf` (que deberá estar instalado). Una ventaja extra es que ambas versiones de `update-eps.sh` serán iguales, no como en el caso anterior en el que había que diferenciar entre vectoriales y de imágenes de bits.

  Por mantener la coherencia, los ficheros deberían renombrarse a algo como `update-pdf.sh`, en cuyo caso habrá que ajustar los *scripts* `updateAll.sh` para modificar su invocación.

- Modificar el fichero `Makefile` principal (en el directorio “raíz” de TeXiS) en varios puntos:

---

[2] *No* se soportan subdirectorios adicionales dentro de los directorios de cada capítulo. Esta restricción se debe al modo en el que los *scripts* buscan los ficheros de imágenes que hay que convertir. Consulta cualquiera de los ficheros `updateAll.sh` dentro de `Imagenes/Vectorial` o `Imagenes/Bitmap` para ver los detalles.

  - Poner como objetivo por defecto a `latex` en lugar de a `pdflatex`. Esto es una mera cuestión de comodidad, y para llevarlo a cabo basta colocar en primer lugar el objetivo de `latex`.
  - Renombrar `fast` a `fastpdflatex` y `fastlatex` a `fast`. De nuevo, esto es una cuestión de comodidad.

- Modificar el guión (*script*) `cleanAll.sh` situado tanto en el directorio `Imagenes/Vectorial` como en `Imagenes/Bitmap`. Ambos son invocados durante la ejecución del objetivo `clean` del `Makefile` principal. En la versión inicial de TeXiS, *se borran* los ficheros `.eps`, dado que son producto de la compilación. Al usar LaTeX, hay que conservarlos, y borrar los ficheros `.pdf` generados en los directorios de las imágenes.

- Modificar el fichero `.cvsignore` o el equivalente en el sistema de control de versiones que se esté usando (si hay alguno) para que se ignoren los nuevos tipos de ficheros generados (`.pdf`) en lugar de los `.eps` que vienen predefinidos en TeXiS. Consulta la sección 4.5 para más información.

6.3.2. Makefile, GlossTeX, y cambio de modo de generación

En la sección 5.2.2 se comentaba que al cambiar el modo de generación del documento desde “depuración” (*debug*) a versión final (*release*) o viceversa, era necesario realizar un `make clean` previo debido al uso de GlossTeX. Aunque en la práctica es suficiente con recordar hacerlo, en esta sección se explican las causas de esta necesidad.

Como se recordará de la sección 5.2.1, cuando se utilizan acrónimos, tras varias etapas termina consiguiéndose un fichero con extensión `.glx` con la descripción de aquéllos que se usaron. En la siguiente compilación del documento, el comando `\printglosstex` se encargará de recoger el contenido de dicho fichero e incrustarlo en esa posición.

Ten en cuenta que la primera vez que se compila el documento, *no* existirá ningún fichero `.glx`; esto es perfectamente legal, dado que `\printglosstex` no se quejará si el fichero *no* existe. Por desgracia, sí generará un error si existe *pero está vacío*.

Cuando en TeXiS el uso de acrónimos está desactivado (lo que ocurre en la generación de depuración), los comandos de GlossTeX como `\acs`, `\acl` etcétera se expanden y no se tienen en cuenta, por lo que en los ficheros intermedios no se informará del uso de ningún acrónimo. Sin embargo, durante el proceso de compilación del documento, el `Makefile` insistirá en realizar los pasos necesarios para la generación de acrónimos (es decir, invocar al ejecutable `glosstex` y a `makeindex`). Éstos desembocarán en la creación de un fichero `.glx` vacío, al no haber ningún acrónimo usado. En principio, esto significa que `\printglosstex` fallará. Afortunadamente, TeXIS *no* añade dicho comando LaTeX al documento cuando los acrónimos están desactivados, lo que evita el problema.

Sin embargo, cuando se activa su generación (al pasar a modo de compilación *release*), TeXIS comienza a incluir el comando. En la primera compilación del documento, por tanto, `\printglosstex` se encontrará que el fichero `.glx` está vacío (de la compilación en *debug* anterior), y fallará. Para solucionarlo, basta en realidad con eliminar dicho fichero; sin embargo, resulta mucho más fácil de recordar (y por tanto práctico) limitarse a realizar un `make clean` que borra, entre otros, el `.glx` por nosotros.

Por otro lado, cuando el documento se ha generado correctamente con los acrónimos (normalmente en modo *release*) y se vuelve a generar sin ellos (en modo *debug*), en la primera compilación LaTeX (o PdfLaTeX) hará uso tanto de los fuentes del documento como de los ficheros auxiliares (`.aux`) generados en la compilación anterior para acelerar el proceso. En esos ficheros se encuentran entradas relativas a los acrónimos que se utilizaron en la última compilación (en *release*). Al hacer ahora la compilación sin acrónimos, no se habrán incluído los paquetes LaTeX que comprenden los comandos de GlossTeX, por lo que LaTeX fallará al no comprenderlos. De nuevo, para solucionar este problema es suficiente con borrar los ficheros `.aux` generados en la compilación anterior; sin embargo resulta mucho más simple realizar un `make clean` con la infraestructura de generación proporcionada por TeXIS.

Notas bibliográficas

Sobre la utilidad `make` hay una gran cantidad de información en Internet. Quizá el lugar de referencia es la página oficial del proyecto de GNU (`http://www.gnu.org/software/make/`), aunque contiene mucha más información de la necesaria para comprender el `Makefile` proporcionado con TeXIS. Por el mero hecho de proporcionar también una referencia impresa, puede consultarse también Mecklenburg (2004).

Parte III

Apéndices

Apéndice A

Así se hizo...

Pones tu pie en el camino y si no cuidas tus pasos, nunca sabes a donde te pueden llevar.

John Ronald Reuel Tolkien, El Señor de los Anillos

Resumen: Este apéndice cuenta algunos aspectos prácticos que nos planteamos en su momento durante la redacción de la tesis (a modo de "así se hizo nuestra tesis"). En realidad no es más que una excusa para que éste manual tenga un apéndice que sirva de ejemplo en la plantilla.

A.1. Edición

Ya indicamos en la sección 3.6 (página 31) que TeXiS está preparada para integrarse bien con emacs, en particular con el modo AucTeX.

Eso era en realidad un síntoma indicativo de que en nuestro trabajo cotidiano utilizamos emacs para editar ficheros LaTeX. Es cierto que inicialmente utilizamos otros editores creados expresamente para la edición de ficheros en LaTeX, pero descubrimos emacs y ha llegado para quedarse (la figura 3.1 mostraba una captura del mismo mientras creábamos este manual). Ten en cuenta que si utilizas Windows, también puedes usar emacs para editar; no lo consideres como algo que sólo se utiliza en el mundo Unix. Nosotros lo usamos a diario tanto en Linux como en Windows.

No obstante, hay que reconocer que emacs *no* es fácil de utilizar al principio (el manual de referencia de Stallman (2007) tiene más de 550 páginas); su curva de aprendizaje es empinada, especialmente si quieres sacarle el máximo partido, o al menos beneficiarte de algunas de sus combinaciones de teclas.

Pero una vez que consigues *no* mover las manos para desplazar el cursor sobre el documento, manejas las teclas rápidas para añadir los comandos LaTeX más utilizados y conoces las combinaciones de AucTeX para moverte por el documento o buscar las entradas de la bibliografía, no cambiarás fácilmente a otro editor.

Si quieres aprovechar emacs, no debes dejar de leer el documento que nos introdujo a nosotros en el modo AucTeX, "*Creación de ficheros LaTeX con GNU Emacs*" (López, 2004).

A.2. Encuadernación

Si has mirado con un poco de atención este manual, habrás visto que los márgenes que tiene son bastante grandes. TeXiS no configura los márgenes a unos valores concretos sino que, directamente, utiliza los que se establecen por defecto en la clase `book` de LaTeX.

Aunque es más o menos reconocido que si LaTeX utiliza esos márgenes debe tener una razón de peso (y de hecho la tiene, se utilizan esos para que el número de letras por línea sea el idóneo para su lectura), cuando se comienza a mirar el documento con los ojos del que quiere verlo encuadernado, es cierto que parecen excesivos. Y empiezas a abrir libros, regla en mano, para medir qué márgenes utilizan. Y reconoces que son mucho más pequeños (y razonables) que el de tu maravilloso escrito. Al menos ese fue nuestro caso.

En ese momento, una solución es *reducir* esos márgenes para que aquello quede mejor. Sin embargo nuestra opción no fue esa. Si tu situación te permite *no* encuadernar el documento en formato DIN-A4, entonces puedes ir a la reprografía de turno y pedir que, una vez impreso, te guillotinen esos márgenes.

Tu escrito quedará entonces en "formato libro", mucho más manejable que el gran DIN-A4, y con unos márgenes mucho más razonables. La figura A.1 muestra el resultado, comparando el tamaño final con el de un folio, que aparece superpuesto.

A.3. En el día a día

Para terminar este breve apéndice, describimos ahora un modo de trabajo que, si bien no utilizamos en su día para la escritura de la tesis, sí hemos utilizado desde hace algún tiempo para el resto de nuestros escritos de LaTeX, incluidos TeXiS y éste, su manual.

Estamos hablando de lo que se conoce en el mundo de la ingeniería del software como *integración contínua* (Fowler, 2006). En concreto, la integración contínua consiste en aprovecharse del servidor del control de versiones para realizar, en cada *commit* o actualización realizada por los autores, una

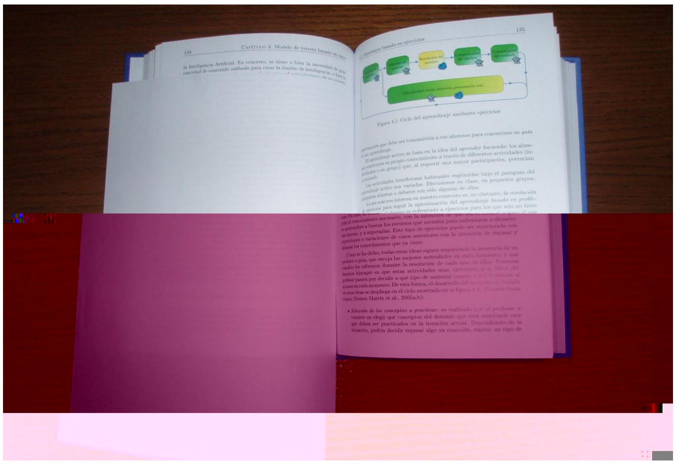


Figura A.1: Encuadernación y márgenes guillotinados

comprobación de si los ficheros que se han subido son de verdad correctos.

En el mundo del desarrollo software donde un proyecto puede involucrar decenas de personas realizando varias actualizaciones diarias, la integración contínua tiene mucha importancia. Después de que un programador realice una actualización, un servidor dedicado comprueba que el proyecto sigue compilando correctamente (e incluso ejecuta los test de unidad asociados). En caso de que la actualización haya estropeado algo, el servidor de integración envía un mensaje de correo electrónico al autor de ese *commit* para avisarle del error y que éste lo subsane lo antes posible, de forma que se perjudique lo menos posible al resto de desarrolladores.

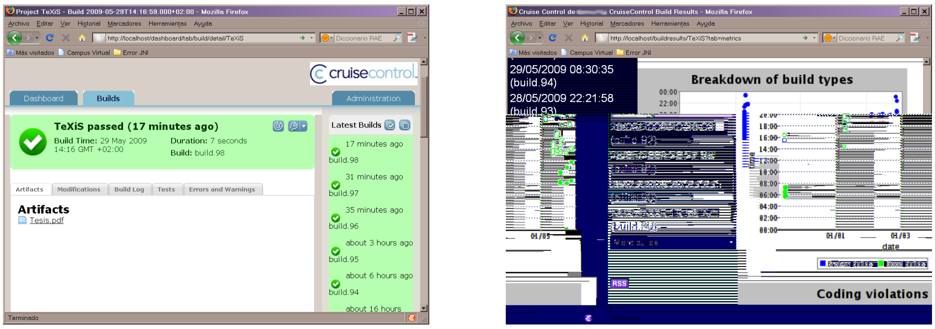
Esa misma idea la hemos utilizado en la elaboración de TeXiS y de este manual. Cada vez que uno de los autores subía al SVN algún cambio, el servidor comprobaba que el fichero maestro seguía siendo correcto, es decir, que se podía generar el PDF final sin errores.

No entraremos en más detalles de cómo hacer esto. El lector interesado puede consultar Gómez-Martín y Gómez-Martín (2010). Como se explica en ese artículo algunas ventajas del uso de esta técnica son:

- Se tiene la seguridad de que la versión disponible en el control de versiones es válida, es decir, es capaz de generar sin errores el documento final.

- Se puede configurar el servidor de integración contínua para que cada vez que se realiza un *commit*, envíe un mensaje de correo electrónico *a todos los autores* del mismo. De esta forma todos los colaboradores están al tanto del progreso del mismo.

- Se puede configurar para que el servidor haga público (via servidor


Project TeXiS - Build 2009-05-29T14:16:59.000+02:00 - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
http://localhost/dashboard/tab/build/detail/TeXiS
cruisecontrol
Dashboard
Builds
Administration
Artifacts
Modifications
Build Log
Tests
Errors and Warnings
Latest Builds
Terminado
Cruise Control de CruiseControl Build Results - Mozilla Firefox
http://localhost/buildresults/TeXiS?tab=metrics
Breakdown of build types
date
Coding violations
RSS

TeXiS passed (17 minutes ago)
Build Time: 29 May 2009 14:16 GMT +02:00
Duration: 7 seconds
Build: build.98

Artifacts
Tesis.pdf

17 minutes ago build.98
31 minutes ago build.97
35 minutes ago build.96
about 3 hours ago build.95
about 6 hours ago build.94
about 16 hours

29/05/2009 08:30:35 (build.94)
28/05/2009 22:21:58 (build.93)

(a) Página de descarga del documento generado

(b) Métricas del proyecto

Figura A.2: Servidor de integración contínua

Web) el PDF del documento (ver figura A.2a). Esto es especialmente útil para revisores del texto como tutores de tesis, que no tendrán que preocuparse de descargar y compilar los `.tex`.

Por último, el servidor también permite ver la evolución del proyecto. La figura A.2b muestra una gráfica que el servidor de integración contínua muestra donde se puede ver la fecha (eje horizontal) y hora (eje vertical) de cada *commit* en el servidor; los puntos rojos representan commits cuya compilación falló.

Bibliografía

> *Y así, del mucho leer y del poco dormir,*
> *se le secó el celebro de manera que vino*
> *a perder el juicio.*
>
> Miguel de Cervantes Saavedra

Abrahams, P. W. *TeX for the Impatient*. Addison Wesley Publishing Company, 1990.

Bautista, T., Oetiker, T., Partl, H., Hyna, I. y Schlegl, E. *Una Descripción de LaTeX 2ε*. Versión electrónica, 1998.

Buckland, M. *Programming Game AI by Example*. Wordware Publishing, Inc., 2005.

Collins-Sussman, B., Fitzpatrick, B. W. y Pilato, C. M. *Version Control with Subversion*. O'Reilly, 2004. ISBN 0-596-00448-6. Disponible en `http://svnbook.red-bean.com/` (último acceso, Octubre, 2009).

Eijkhout, V. *TeX by Topic, a TeXnician's Reference*. Addison-Wesley, 1991.

Flynn, P. *A beginner's introduction to typesetting with 11.TJF1.-1622.2TdATJF1.1Tf.6-2.2TdTTJ.-2.3*

M.

GÓMEZ-MARTÍN, P. P. *Modelo de enseñanza basado en casos: de los tutores inteligentes a los videojuegos*. Phd, Universidad Complutense de Madrid, 2008b.

KALDERON, M. E. LaTeX and Subversion. *The PracTeX Journal*, vol. 2007(3), 2007.

KNUTH, D. E. *The TeX book*. Addison-Wesley Professional., 1986.

KRISHNAN, E., editor. *LaTeX Tutorials. A primer*. Indian TeX Users Group, 2003.

LAMPORT, L. *LaTeX: A Document Preparation System, 2nd Edition*. Addison-Wesley Professional, 1994.

LÓPEZ, J. A. *Creación de ficheros LaTeX con GNU Emacs*. Versión electrónica, 2004. Disponible en `http://ctan.org/get/info/spanish/guia- tx/guia- tx.pdf` (último acceso, Octubre, 2009).

LÓPEZ, J. A. *Guía* casi *completa de BibTeX*. Versión electrónica, 2006. Disponible en `ftp://ftp.ctan.org/tex-archive/info/spanish/guia-bibtex/guia-bibtex.pdf` (último acceso, Octubre, 2009).

MECKLENBURG, R. *Managing Projects with GNU Make, 3rd edition*. O'Reilly Media, Inc, 2004. ISBN 0596006101.

MITTELBACH, F., GOOSSENS, M., BRAAMS, J., CARLISLE, D. y ROWLEY, C. *The LaTeX Companion*. Addison-Wesley Professional, segunda edición, 2004.

OETIKER, T., PARTL, H., HYNA, I. y SCHLEGL, E. *The Not So Short Introduction to LaTeX 2ε*. Versión electrónica, 1996.

PATASHNIK, O. BibTeXing. 1988a. Disponible en `ftp://ftp.ctan.org/tex-archive/biblio/bibtex/contrib/doc/btxdoc.pdf` (último acceso, Octubre, 2009).

PATASHNIK, O. Designing BibTeX styles. 1988b. Disponible en `ftp://ftp.ctan.org/tex-archive/biblio/bibtex/contrib/doc/btxhak.pdf` (último acceso, Octubre, 2009).

SCHARRER, M. Version control of LaTeX documents with svn-multi. *The PracTeX Journal*, vol. 2007(3), 2007.

STALLMAN, R. M. *GNU Emacs Manual for Version 22, 16th Edition*. Free Software Foundation, 2007. Disponible en `http://www.gnu.org/software/emacs/manual/emacs.pdf` (último acceso, Octubre, 2009).

THOUGHTWORKS, I. Cruise control. 2001. Disponible en `http://cruisecontrol.sourceforge.net/index.html` (último acceso, Octubre, 2009).

VESPERMAN, J. *Essential CVS*. O'Reilly, 2003. ISBN 0-596-00459-1.

WIKIPEDIA (LaTeX). Entrada: "LaTeX". Disponible en `http://es.wikipedia.org/wiki/LaTeX` (último acceso, Octubre, 2009).

ZIEGENHAGEN, U. LaTeX document management with Subversion. *The PracTeX Journal*, vol. 2007(3), 2007.

Lista de acrónimos

CVS	*Control Version System*, Sistema de Control de Versiones
GPL	*General Public License*, Licencia Pública General de GNU
PC^2	*Programming Contest Control*

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes