Guía rápida para el nuevo usuario de LaTeX

Eugenio M. Fedriani Martel (Coord.)

Sevilla, julio de 2004.

AUTORES:

Francisco J. Blancas Peral

María Cortada García

Eugenio M. Fedriani Martel

Alfredo García Hernández-Díaz

Irene García Selfa

Paula González Rodríguez

Sandra González Salas

Mª del Pilar Moreno Navarro

Raquel Rafael Arenas

Mª Isabel Sanz Domínguez

Ángel F. Tenorio Villalón

María M. Vega Quirós

Índice general

Capítulo 1

Nociones básicas de LaTeX y su funcionamiento

1.1. Antecedentes históricos

Donald Ervin Knuth creó en 1978 un sistema de composición de textos de alta calidad llamado TeX. En realidad, el origen de TeX se remonta a marzo del año 1977, cuando Knuth recibió las pruebas de imprenta de la segunda edición del segundo volumen de su libro *The Art of Computer Programming*. Dichas pruebas de imprenta le causaron tan mala sensación que las llegó a calificar de tipográficamente incorrectas. Visto lo cual Knuth se decidió a crear un sistema de composición de textos y unos archivos de estilo tipográficos para emplearlos en textos de alta calidad y, más concretamente, pensando en aquellos textos en los que aparecían expresiones matemáticas. En julio de ese mismo año concluyó dos informes de uso personal en los que ya estaban encerrados el germen de TeX y Metafont. Un año más tarde Knuth participó en el encuentro anual de la American Mathematical Society (AMS) como conferenciante invitado y expuso el proyecto de investigación que daría lugar a TeX en 1978. En septiembre de ese mismo año, Standford editaba el primer manual de TeX. Esto se debió a que Knuth buscó, además del apoyo de la AMS, el de la editorial Addison-Wesley para llevar a cabo su proyecto.

El creador de TeX, lleva el perfeccionismo hasta el extremo; a modo de ejemplo, publica en su página web, un mensaje de "aviso importante" en el que dice que si tu sistema produce el símbolo delta minúscula de forma ligeramente diferente a

$$\delta$$

debes actualizar las fuentes.

Leslie Lamport, en 1982, creó un paquete de macros para TeX, llamado LaTeX (proveniente de LAmport TeX) y que facilita el uso del TeX sin disminuir su potencia. LaTeX proporciona una serie de órdenes para describir la estructura del documento, con el fin de que el usuario se preocupe más del contenido del documento que de su presentación.

Distintos usuarios de TEX y LATEX han venido colaborando con sus aportaciones, inventando herramientas para facilitar el uso de estos sistemas. Para usuarios hispanohablantes, la página web [3] proporciona información sobre versiones de TEX, congresos y otras noticias de interés sobre el uso de TEX y LATEX en Español.

Uno de los enlaces que aparece en la página web [3] responde a la pregunta "¿Qué es TEX?" con las siguientes afirmaciones que pueden resultar aclaradoras: "No es un procesador de texto. No es un programa de maquetación. Es un sistema de fotocomposición."

1.2. Procesadores de texto

Hay dos tipos de procesadores de texto o programas de maquetación. Los procesadores WYSIWYG (What You See Is [all] What You Get), como Microsoft Word[1] o Corel WordPerfect[2], en los que se obtiene lo que se está viendo mientras se escribe; y los sistemas de fotocomposición automatizados, como el LATEX, en los que se necesita un compilador para ver el resultado final del documento.

Los principales incovenientes de LATEX son, por una parte, la necesidad de un proceso lento de aprendizaje y, por otra, la incomodidad para diseñar. Sin embargo, algunas de sus muchas ventajas son las siguientes: es más rápido en la producción de documentos, el resultado final tiene una calidad profesional, es gratuito y existen versiones para ordenadores poco potentes y con pocos recursos (podemos escribir en cualquier procesador de textos, como por ejemplo, el Microsoft Bloc de Notas[3] de Microsoft Windows[4], las órdenes necesarias para la producción del documento, ocupando muy poca memoria).

1.3. Aspectos generales de TEX y LATEX

Para escribir en TEX y LATEX, solo deberíamos utilizar los caracteres estándar del código ASCII, es decir, los generados por los siete primeros bits. Para escribir símbolos como la tilde o la eñe, disponemos de una serie de comandos, según se verá.

También existen editores especiales (como WinEdt[5] o WinShell[6]) que ayudan a generar estos caracteres.

Podemos diferenciar tres fases en la producción de un documento: preparación, procesado e impresión.

En la fase de preparación, se crea un archivo binario de texto (archivo.tex) con las órdenes y comandos necesarios para la interpretación del documento. Al utilizar

[1] Microsoft Word ©1983–2004 es marca registrada de Microsoft Corporation.
[2] Corel WordPerfect ©1979(1996)–2004 es marca registrada de Corel Corporation.
[3] Microsoft Bloc de Notas ©1981–2004 es marca registrada de Microsoft Corporation.
[4] Microsoft Windows ©1981–2004 es marca registrada de Microsoft Corporation.
[5] WinEdt ©1993–2004 es marca registrada de Aleksander Simonic.
[6] WinShell ©1999-2004 es marca registrada de Ingo H. de Boer.

solamente los siete primeros bits, el archivo.tex ocupa muy poco espacio y puede ser transportado y visto en otro ordenador de diferentes características, sin que cambie el contenido del archivo ni se modifique ningún carácter.

En la fase de procesado, obtenemos información sobre las características del documento y los posibles errores cometidos al escribir en TEX o LATEX. Tenemos la posibilidad de volver a la fase de preparación para subsanar dichos errores o realizar cualquier otro cambio.

Por último, vemos el resultado (no necesariamente en papel) en la fase de impresión. Es ahora cuando podemos pasar a otros formatos que hacen visible el documento tal y como aparecería en papel. Normalmente usaremos los archivos de extensión .DVI, .PS o .PDF. También es posible volver a la fase de preparación para realizar los cambios que estimemos oportunos, una vez hemos conocido el resultado.

De los símbolos especiales que se utilizan al escribir en TEX o LATEX, debemos destacar: la barra invertida o backslash (\), que significa que vamos a escribir un comando; el símbolo del dólar (`$`), que se usa para abrir y cerrar fórmulas; las llaves (`{}`), que sirven para delimitar partes del documento.

Los entornos nos permiten diferenciar el tratamiento de diferentes elementos en un documento: un título, una tabla, un dibujo, una enumeración o una fórmula. En la mayoría de ellos hay que indicar dónde empiezan y dónde terminan, como por ejemplo:

```
\begin{table}
⋮
\end{table}
```

Como TEX se parece más a un lenguaje de programación que a un procesador de textos, podemos encargar al programa que se encarge de tareas como crear páginas web (de hecho, `html` surgió del TEX), crear un índice a partir de un documento, crear una bibliografía, hacer referencias cruzadas, etc.

1.4. Otros aspectos de TEX y LATEX

Para encontrar información sobre LATEX, además de en las páginas de Knuth [10] y Lamport [12], en la de Bausela [1] podemos ver información útil para el usuario hispanohablante que se inicia en TEX.

Para empezar a manejar LATEX, lo primero que hay que hacer es estructurar el documento; es decir, tenemos que saber qué tipo de texto queremos, qué partes va a tener y qué entornos vamos a utilizar. Esto hará posible que el documento tenga una estructura formal lógica muy apreciada en documentos científicos.

Hay tres servidores FTP oficiales o CTAN (*Comprehensive TEX Archive Network* o *Red de Archivos de TEX Completo*) que almacenan prácticamente todo lo relacionado con TEX. Sus *nodos* son:

- `ftp.dante.de` (Alemania)
- `ftp.tex.ac.uk` (Reino Unido)
- `ctan.tug.org` (Estados Unidos)

En España existe una réplica o espejo del CTAN en `ftp.rediris.es`, donde se pueden buscar archivos, información o enlaces con páginas que tienen que ver con LaTeX, aunque no se actualiza con la misma rapidez que los nodos del CTAN.

Desde la página de Bausela [1] se puede bajar e instalar MiKTeX, que es una herramienta muy útil para trabajar en Windows con TeX y los programas relacionados. Está formado por más de mil paquetes que se actualizan periódicamente.

También podemos descargar MiKTeX desde su página web [15], que nos remite a CTAN.

1.5. Programas convenientes para el uso de TeX

Además de LaTeX, necesitamos un editor de textos, que puede ser el Bloc de Notas, Word o cualquier otro; aunque los mejores editores de texto (de los que están preparados para LaTeX) son WinShell (gratuito) y WinEdt. Este último es posiblemente el editor de texto más potente del mundo y podemos encontrar versiones de prueba en su página web [17].

Por otra parte, necesitamos un visor para ver el aspecto que tiene el documento que estamos realizando. Podemos usar un visor de archivos DVI, PS o PDF. De los tres tipos hay programas gratuitos.

1.6. Otros programas auxiliares para el uso de TeX

A lo largo de este manual se irán introduciendo otros programas que resultan interesantes para el que quiera usar LaTeX. Cabe destacar la gran utilidad de diversos paquetes incluidos en MiKTeX, entre los que podemos encontrar BibTeX, que es el más usado para generar bibliografías.

Mayura Draw[7] es posiblemente la herramienta de diseño gráfico más potente que encontramos, aunque también podemos usar TeXPict[8] o LaTeXCAD[9] para generar dibujos. Con ambos programas se puede ver el dibujo que se está realizando, pero además genera un archivo ASCII que puede incorporarse sin mayor dificultad al archivo TeX que estemos escribiendo.

[7] Mayura Draw ©1993–2004 es marca registrada de Mayura Software.
[8] TeXPict ©1999–2004 es marca registrada de Ramón Ribó.
[9] LaTeXCAD © 1998–2004 es marca registrada de John Leis.

Otros programas auxiliares interesantes son TEXaide[10], que traduce (una a una) fórmulas de Word a TEX y viceversa; y Mathematica[11] o Maple[12], que son programas de cálculo simbólico a los que le podemos pedir que las salidas las exporten en TEX.

1.7. Producción de un documento

Al escribir un documento, como se ha dicho, lo que hacemos es generar un archivo.tex, con sus órdenes y comandos particulares. LATEX es el encargado de compilar dicho archivo, generando otros, como son el archivo.dvi, el archivo.log o el archivo.aux.

El archivo.dvi es como un archivo gráfico. Se puede ver su aspecto e imprimir, pero no se puede modificar directamente.

En el archivo.log podemos encontrar una lista con los errores y aciertos que se han cometido. Hay cierto tipo de errores que LATEX se encarga de reparar automáticamente, pero de otros nos avisa especialmente durante la compilación.

La información que hace falta para generar los archivos .DVI se almacena en el archivo.aux. Por ejemplo, guarda la información necesaria para hacer un índice, de manera que la primera vez que compilamos se crea el archivo.aux y es a partir de la segunda vez cuando tomará esa información y creará el índice.

Una vez generados todos los archivos anteriores, podemos generar uno nuevo con extensión .ps, gracias al programa dvips.exe. A partir de éste, se puede obtener el archivo.pdf, con la ayuda del programa Adobe Acrobat[13] o de programas como ps2pdf.exe.

También existe la opción de pasar de un archivo de texto (.TEX) a otro .PS o .PDF directamente, utilizando los programas tex2ps.exe y tex2pdf.exe, respectivamente.

Veamos un primer ejemplo sencillo de lo que sería un documento de texto escrito en LATEX:

```
\documentclass{article}
    \begin{document}
        $$ \int_0^\pi {\sqrt[3]{x} \over \lambda} \, dx $$
    \end{document}
```

Y en el correspondiente archivo.dvi obtendríamos la siguiente vista:

$$\int_0^{\pi} \frac{\sqrt[3]{x}}{\lambda}\, dx$$

[10] TEXaide ©1990–2004 es marca registrada de Design Science, Inc.
[11] Mathematica ©1988–2004 es marca registrada de Wolfram Research.
[12] Maple ©1981–2004 es marca registrada de Waterloo Maple Inc.
[13] Adobe Acrobat ©1987–2004 es marca registrada de Adobe Systems Inc.

Más adelante iremos viendo para qué sirve cada comando utilizado. De momento este ejemplo podría servirnos para comparar la calidad de la salida y la poca memoria ocupada en relación con otros procesadores de texto.

Capítulo 2

Tipos de documento. Partes de un documento. Fórmulas matemáticas

2.1. Repaso al funcionamiento del programa

Como ya sabemos, LaTeX es un sistema de composición tipográfica que toma como punto de partida para la generación de un documento un archivo.tex, es decir, un archivo de texto en formato ASCII. Éste puede ser generado con cualquier editor de textos y deberá contener tanto el texto que se desea aparezca en el documento como las "instrucciones" o "comandos", con los que LaTeX reconoce la forma final en la que debe disponer los distintos elementos del documento. En las órdenes de LaTeX se distinguen las letras mayúsculas de las minúsculas, tomando habitualmente uno de los dos siguientes formatos:

1. Los compuestos por la barra \, llamada *backslash*, seguida de un nombre compuesto exclusivamente por letras. El nombre del comando concluye con el primer carácter distinto de una letra, y que habitualmente es un espacio en blanco. Un ejemplo de comando de este formato es el empleado para escribir el nombre del programa LaTeX: `\LaTeX`.

2. Otro tipo de comandos de LaTeX se obtienen escribiendo el carácter \ seguido de un carácter que no es una letra y se emplean para escribir, por ejemplo, los siguientes carácteres especiales de LaTeX: {, }, $, #, &, ~, _, ^, %. Los respectivos comandos empleados en LaTeX para dichos caracteres especiales son: `\{`, `\}`, `\$`, `\#`, `\&`, `\~`, `\_`, `\^`, `\%`.

Dentro de estos formatos, algunas instrucciones presentan además argumentos obligatorios que deben aparecer para su correcto funcionamiento. Estos argumentos aparecerán escritos siempre entre dos llaves, `{`*Argumento Obligatorio*`}`, tras la instrucción correspondiente. Un ejemplo de estos comandos es `\textsl{`*Texto*`}`, que escribe el texto que aparece en su argumento obligatorio en *letra cursiva*.

Otras órdenes, además, pueden llevar argumentos opcionales que normanlmente deben añadirse escritos entre corchetes, [*Argumento Opcional*] , tras el nombre de la instrucción y antes de escribir los argumentos obligatorios. Para ver algunos ejemplos de este tipo de comandos, en la Sección 2.2, nos dedicaremos a la estructura del archivo de entrada.

Una vez editado el archivo.tex, éste se procesa y se va corrigiendo para finalmente obtener el documento a imprimir. Este proceso se realiza en LaTeX al ir compilando el archivo.tex que va generando otros archivos adicionales que le son necesarios a LaTeX para la producción final del documento. Entre ellos ya hemos destacado los siguientes:

1. archivo.dvi: es un archivo "gráfico" que no puede modificarse y donde se va mostrando el resultado final de lo que se va escribiendo en el archivo.tex.

2. archivo.log: es un archivo donde LaTeX va escribiendo los errores que se van cometiendo en la edición del texto.

3. archivo.aux: en este archivo LaTeX recoge la información que necesitará para generar otros archivos como, por ejemplo, un archivo.dvi correcto.

Finalmente, con el archivo.dvi terminado se puede pasar a otro formato como por ejemplo el archivo.pdf, de manera que pueda ser leído en cualquier equipo e imprimido sin problemas por usuarios que no conozcan en absoluto TeX ni LaTeX.

2.2. Estructura de un archivo de entrada

Cuando LaTeX procesa un archivo de entrada, espera que éste siga una determinada estructura.

Todo archivo de entrada debería comenzar con la siguiente orden:

\documentclass{Argumento}.

Dicha orden indica a LaTeX la clase de documento que se pretende crear. Tras ella, se abre un espacio donde se pueden incluir órdenes que influirán sobre el estilo del documento entero o cargar paquetes que añadirán nuevas propiedades al sistema LaTeX. Para cargar los citados paquetes se utiliza la instrucción \usepackage{*Paquete*}.

A todo este espacio se le denomina *preámbulo del documento*. Una vez se ha finalizado el trabajo de configuración del documento en el preámbulo, se escribe el comando \begin{document}, con la que se inicia la parte del documento correspondiente al contenido del texto, donde se desarrolla el mismo mezclado con algunas instrucciones útiles de LaTeX. Una vez finalizado el texto, se debe acabar el documento con la orden \end{document}. LaTeX ignorará cualquier cosa que se ponga tras esta instrucción.

De esta forma, cuando se procesa un archivo.tex lo primero que hay que indicarle a LaTeX es el tipo de documento que se quiere elaborar. Eso ya sabemos que debemos hacerlo con la siguiente orden: `\documentclass[`*Opciones*`]{`*Clase*`}`.

Este comando tiene dos tipos de argumentos. En primer lugar, el argumento obligatorio *Clase* donde debe indicarse el tipo de documento que vamos a crear. Entre los tipos de documentos que se pueden crear destacan como los más usuales:

1. `article`: se utiliza para elaborar artículos de revistas especializadas, ponencias, trabajos, seminarios, informes pequeños, etc.

2. `report`: normalmente se utiliza para crear informes mayores que constan de más de un capítulo, proyectos fin de carrera, tesis doctorales o libros pequeños, entre otros.

3. `book`: se emplea para crear libros u otros documentos a doble cara de características similares a libros.

4. `slide`: se usa para elaborar transparencias.

Además, este comando inicial presenta argumentos opcionales que sirven para personalizar el comportamiento del tipo de documento que estamos elaborando. Estas opciones podrán ser varias y todas ellas irán separadas por comas. Las opciones más comunes que suelen indicarse son las siguientes:

a) `10pt`, `11pt`, `12pt` ...: establecen el tamaño de la letra con la que se va a escribir el documento por defecto. Si no se indica, se supone que es de `10pt`.

b) `a4paper`, `letterpaper` ...: definen el tamaño del papel en el que se va a escribir el texto. Si no se indica nada, LaTeX toma por defecto el tamaño `letterpaper`. Aparte de los anteriores se pueden utilizar otros como son `a5paper`, `b5paper`, etc.

c) `twocolum`: con esta opción se le indica a LaTeX que componga el documento en 2 columnas.

d) `landscape`: con esta opción el documento final se escribe en forma apaisada.

d) `twoside`, `oneside`: especifica si se debe generar el documento a una o dos caras. Si no se especifica nada, los documentos tipo `article` y `report` son a una cara y los de clase `book` a dos caras.

De esta forma, un archivo de entrada para un documento de LaTeX, podría empezar con `\documentclass[12pt,landscape,a4paper]{article}`. Esta orden le indica a LaTeX que componga el documento como un artículo, con una letra de tamaño 12pt, que disponga el texto en forma apaisada a una cara en un papel del tamaño DIN-A4.

Tras esta orden tendríamos que indicarle a LaTeX qué aspecto final y qué configuración deseamos para nuestro documento, todo ello en el citado preámbulo. En el presente capítulo solo nos vamos a centrar en el análisis del contenido del texto, dejando para capítulos posteriores el análisis del preámbulo.

2.3. Composición del texto

2.3.1. Saltos de línea y de página. Espacio entre palabras

Por defecto, LaTeX inserta los saltos de línea y los espacios entre palabras optimizando el contenido de los párrafos enteros. Si es necesario, también introduce guiones dividiendo las palabras que no encajan bien al final de cada renglón. No obstante, el modo en el que LaTeX compone los párrafos dependerá del tipo de documento que se esté elaborando.

A pesar de todo ello, es posible indicarle a LaTeX que incluya un salto de línea allí donde se desea. Para ello se pueden utilizar comandos como los siguientes: `\\` o `\newline`. Con estos comandos, LaTeX termina una línea y pasa a la siguiente sin comenzar un párrafo nuevo. Un resultado similar puede conseguirse comenzando el nuevo párrafo con `\par` o dejando dos espacios en blanco entre línea y línea.

Si, además, a los comandos anteriores se les añade al final un asterísco, como con `\\*`, se le prohibe a LaTeX que se produzca un salto de página tras el salto de línea.

También puede indicársele a LaTeX que cambie de página cuando se desee, utilizando el comando `\newpage`.

De la misma forma en la que se le puede obligar a LaTeX a crear espacios entre líneas, existen también comandos que permiten variar el espacio que por defecto establece entre palabras. Así, si utilizamos el comando `\,` LaTeX deja un espacio pequeño entre palabras, como se observa en el siguiente ejemplo:

```
Nunca podremos olvidar este \, maravilloso curso pr\'actico de \LaTeX.
```
Nunca podremos olvidar este maravilloso curso práctico de LaTeX.

De la misma forma, también podemos introducir espacios negativos entre palabras utilizando el comando `\!` como se observa en el siguiente ejemplo:

```
El comando que permite introducir espacios $\!\!\!$ negativos solo
funciona en modo ma\-tem\'a\-ti\-co, como observamos.
```
El comando que permite introducir espacios negativos solo funciona en modo matemático, como observamos.

2.3.2. Caracteres especiales o símbolos

Los símbolos siguientes son *caracteres especiales* que tienen un significado especial para LaTeX, de tal forma que aunque se pueden escribir a través de los siete primeros bits, si lo introducimos directamente en el archivo de entrada es muy probable que

no aparezcan reflejados en el texto e incluso, que dé un error en la compilación del archivo o que forcemos a LaTeX a realizar cosas que no deseamos. Por ello, para que aparezcan apropiadamente en el documento final generado con LaTeX hay que introducirlos predecidos del carácter \, como se vio en la Sección 2.1.

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

2.3.3. Entornos

La forma de indicarle a LaTeX que se quiere realizar algo especial con una determinada parte del texto es a partir de los denominados *entornos*. Los entornos son una especie de "grupos" de comandos que se presentan normalmente con la siguiente estructura:

\begin{*NombreEntorno*}
Texto
\end{*NombreEntorno*}

de tal forma que el efecto de la orden del entorno se aplica al texto que se encuentra entre el inicio y el final del entorno.

Un ejemplo de este tipo de entornos es el que ya hemos utilizado para crear un documento:

\begin{document}
Contenido del documento
\end{document}

También se puede introducir un nuevo entorno dentro de otro, debiéndose de tener mucho cuidado con la secuencia que inicia y cierra cada entorno. De esta forma, dentro de un mismo documento podemos utilizar múltiples entornos, pero no se puede terminar un entorno sin terminar también todos los que se han iniciado después de él:

\begin{*NombreEntorno1*}
Texto
\begin{*NombreEntorno2*}
Texto
\end{*NombreEntorno2*}
Texto
\end{*NombreEntorno1*}

Otro ejemplo sencillo de este tipo de entornos lo constituye el entorno `center` que genera un texto centrado. Si no se introducen saltos de línea para pasar al siguiente renglón, LaTeX lo hará automaticamente donde lo crea más conveniente. Veamos un ejemplo:

```
\begin{center}
Este texto est\'a \\ centrado mediante el entorno {\tt center}
\end{center}
```

Este texto está
centrado mediante el entorno `center`

Como vemos, LaTeX no fuerza a que todas las líneas tengan la misma longitud.

De igual modo, también se pueden utilizar otros entornos similares como son `flushleft` y `flushright`, que producen párrafos justificados a izquierda y derecha respectivamente, sin nivelar el otro borde.

```
\begin{flushleft}
este texto est\'a justificado\\
a la izquierda utilizando el entorno\\
{\tt flushleft}
\end{flushleft}
```

este texto está justificado
a la izquierda utilizando el entorno
`flushleft`

```
\begin{flushright}
este texto est\'a justificado\\
a la derecha
utilizando el entorno\\
{\tt flushright}
\end{flushright}
```

este texto está justificado
a la derecha utilizando el entorno
`flushright`

Otro entorno muy común que sigue el mismo formato anterior es el entorno `quote`, que sirve para realizar citas pequeñas y ejemplos y para resaltar ciertas oraciones. Veamos un ejemplo de cómo actúa este entorno.

```
Este es un ejemplo que nos ense\~na:
\begin{quote}
c\'omo debemos realizar una peque\~na cita o resaltar una
oraci\'on
\end{quote}
utilizando el entorno {\tt quote}.
```

Este es un ejemplo que nos enseña:

> *cómo debemos realizar una pequeña cita o resaltar una oración*

utilizando el entorno `quote`.

Otro ejemplo de este tipo de entornos es `itshape`, cuya utilidad estriba en escribir en letra cursiva todo el texto sobre el que actúa, tal y como se muestra en el siguiente ejemplo.

```
\begin{itshape}
Este es un ejemplo de como act\'ua el entorno {\tt itshape}
\end{itshape}
```

Este es un ejemplo de como actúa el entorno `itshape`

A diferencia de los entornos mostrados en los ejemplos anteriores, existen otros que poseen una estructura diferente, del tipo: `\NombreEntorno{`*Argumento*`}`.

En este caso, la acción del entorno recae sobre el argumento obligatorio que se escribe a continuación entre llaves. Un ejemplo claro de este tipo de entornos es el entorno `textit`, que sirve para escribir un texto en cursiva.

```
\textit{Realizar un curso de \LaTeX \, es lo mejor que te puede
pasar a lo largo de tu vida acad\'emica}
```

Realizar un curso de LaTeX es lo mejor que te puede pasar a lo largo de tu vida académica

Otro entorno que también tiene la estructura anterior es el entorno `underline`, que sirve para subrayar un determinado texto. El texto que se desea subrayar será el que aparecerá como argumento obligatorio escrito entre llaves. Si éstas son obviadas, el entorno actuará sobre el siguiente carácter que aparezca. Vamos a ver cómo funciona el citado entorno con el siguiente ejemplo.

```
\underline{Vamos a subrayar este texto utilizando el entorno
{\tt underline}.}\\
De igual forma si no utilizamos las llaves act\'ua sobre el
siguiente \underline car\'acter.
```

Vamos a subrayar este texto utilizando el entorno `underline`.
De igual forma si no utilizamos las llaves actúa sobre el siguiente carácter.

Por último, un tercer formato que suelen presentar algunos entornos es el que utiliza exclusivamente un comando y, por eso, se ajusta al siguiente: `{\NombreEntorno` *Argumento*`}`. En este caso el nombre del entorno aparece junto al argumento sobre el que actúa en un mismo espacio acotado por llaves que actúan de delimitadores.

Ejemplo de este tipo de entornos lo constituye `it`, que nos permite escribir un determinado texto en letra cursiva (por tercera vez en esta Sección). Para ver como funciona, revisemos el siguiente ejemplo.

```
Durante estas dos semanas que ha durado el curso, {\it las grandes
ense\~nanzas del profesor} han hecho de nosotros unos expertos en
\LaTeX.
```

Durante estas dos semanas que ha durado el curso, *las grandes enseñanzas del profesor* han hecho de nosotros unos expertos en LaTeX.

Otro ejemplo lo constituye el entorno `bf`, que nos permite escribir un determinado texto en letra negrita. Para ver cómo funciona se muestra el siguiente ejemplo:

```
El curso pr\'actico de \LaTeX \, ha sido {\bf muy entretenido e
interesante} a pesar de su horario.
```

El curso práctico de LaTeX ha sido **muy entretenido e interesante** *a pesar de su horario.*

2.3.4. Comentarios

Una de las grandes ventajas que propicia el escribir en LaTeX es que existe la posibilidad de insertar a lo largo del texto ciertos comentarios que no aparecerán en el texto finalmente impreso. Para ello se escribe el símbolo % delante del texto que constituye el comentario. Cuando LaTeX encuentra el carácter % al procesar el archivo.tex, ignora el resto de la línea, tal y como se observa en el siguiente ejemplo.

```
El curso pr\'actico de \LaTeX \, ha sido impartido durante el mes de
mayo de 2004. %a partir de ahora estamos introduciendo un comentario.
```

El curso práctico de LaTeX ha sido impartido durante el mes de mayo de 2004.

2.3.5. Fórmulas matemáticas

A la hora de emplear LaTeX para escribir, existen tres modos de escritura. El primero de ellos es el *modo texto* que es en el que nos hemos estado moviendo hasta este momento. Pero LaTeX fue creado con el fin de facilitar la elaboración de textos científicos y, más concretamente, matemáticos. Pues bien, cuando se quiere escribir una fórmula matemática con LaTeX, debemos avisarle de que en ese momento vamos a dejar de escribir un texto y pasamos a introducir elementos del lenguaje matemático. Esto lo llevamos a cabo introduciendo las fórmulas en *modo matemático*. El objetivo de las siguientes líneas será introducir al lector en cómo se trabaja en modo matemático.

En primer lugar, una fórmula puede ser escrita en **modo texto**, es decir, escribiendo una fórmula exactamente igual que se escribe cualquier parte del texto, tal y como se observa en el siguiente ejemplo.

```
Ahora en estas l\'{\i}neas vamos a escribir una f\'ormula que dice
que y = 3x + 22.
```

Ahora en estas líneas vamos a escribir una fórmula que dice que y = 3x + 22.

Por otra parte, una fórmula puede ser escrita en **modo matemático**. Para activar el modo matemático tenemos, en principio, dos opciones:

a) *Dentro de un párrafo*, es decir, la fórmula se escribe entre dos signos dólar, $*Fórmula*$ y aparece en medio del párrafo.

b) *En forma expandida*, es decir, la fórmula se escribe entre dos pares de signos dólar, $$*Fórmula*$$ y aparece centrada y fuera del párrafo.

Veamos algunos ejemplos que nos muestren las diferencias que existen entre las distintas formas de escribir las fórmulas.

```
En estas l\'{\i}neas vamos a escribir una f\'ormula como es $y =
3x + 22$ utilizando el modo matem\'atico dentro de un p\'arrafo.
```

En estas líneas vamos a escribir una fórmula como es $y = 3x + 22$ *utilizando el modo matemático dentro de un párrafo.*

```
Como observamos, el resultado es diferente si la misma f\'ormula
$$y = 3x +22 $$ se escribe utilizando el modo matem\'atico en
forma expandida.
```

Como observamos, el resultado es diferente si la misma fórmula

$$y = 3x + 22$$

se escribe utilizando el modo matemático en forma expandida.

Observando los ejemplos anteriores, podemos destacar que existen ciertas diferencias en la utilización del modo matemático y el modo texto para escribir la fórmula. Entre ellas podemos destacar las siguientes:

- En modo matemático, los espacios en blanco y los cambios de línea no tienen ningún significado para LaTeX. Todos los espacios son determinados automáticamente por LaTeX en función de la lógica de la expresión matemática.

- A diferencia de lo que ocurre en el modo texto, cuando utilizamos el modo matemático, cada letra utilizada es considerada como una variable, apareciendo escrita en cursiva y sin rodear de espacios adicionales.

Centrándonos ahora en el modo matemático, observamos que existen diferencias importantes cuando la fórmula es insertada en el párrafo de cuando está escrita en forma expandida.

Como se observa en los ejemplos anteriores, cuando escribimos una fórmula en forma expandida, ésta aparece en el texto en un párrafo nuevo y centrada en el

mismo. Además, a diferencia de lo que ocurre cuando la escribimos insertada en el párrafo, los componentes de la misma presentan un tamaño distinto puesto que, en el primer caso, LaTeX adapta el tamaño de las mismas al espacio disponible en la línea del párrafo donde aparece.

Cuando escribimos en modo matemático, la mayoría de las instrucciones solo afectan al carácter siguiente a las mismas. De esta forma, cuando deseemos que una instrucción actúe sobre varios carácteres, entonces éstos deben agruparse empleando delimitadores, que habitualmente son unas llaves. Por ejemplo, si queremos escribir la expresión

$$a^{x+y}$$

debemos escribir la siguiente orden: `$$a^{x+y}$$`

Los *elementos* más comunes que suelen aparecer en las fórmulas matemáticas son los siguientes:

a) ***Exponentes.*** Para indicar a LaTeX que eleve una determinada cantidad a un exponente dado, debemos emplear el carácter `^`. Así, por ejemplo, si queremos escribir la expresión x^{3+y} debemos utilizar la orden `$x^{3+y}$`.

b) ***Subíndices.*** Para indicar a LaTeX que escriba una determinada variable con un subíndice dado, hay que emplear el carácter `_`. De esta forma, si por ejemplo queremos escribir la expresión x_{y+z} debemos escribir la siguiente instrucción: `$x_{y+z}$`.

c) ***Signo de la raíz cuadrada.*** Para escribir en LaTeX este símbolo debemos introducir la instrucción siguiente: `\sqrt{`*Radicando*`}`. Si lo que queremos es escribir la raíz n-ésima, entonces deberemos añadir a la instrucción anterior un argumento opcional donde indiquemos el grado de la raíz: `\sqrt[n]{`*Radicando*`}`. Veamos algunos ejemplos:

`$\sqrt{x^3}$` $\sqrt{x^3}$ `$\sqrt[5]{x^2 + y}$` $\sqrt[5]{x^2 + y}$

Como vemos en los ejemplos, LaTeX elige automáticamente el tamaño del signo de la raíz, adaptándola al tamaño del radicando.

d) ***Quebrado o fracción.*** En LaTeX existen distintas instrucciones que nos permiten escribir una fracción. La orden que se utiliza más comúnmente para ello es `\frac{`*Numerador*`}{`*Denominador*`}`. Como vemos, esta orden presenta dos argumentos obligatorios, el primero de ellos para indicar el valor del numerador y el segundo para el denominador.

Otra instrucción de uso muy común es la que presenta el siguiente formato: `{`*Numerador*`\over`*Denominador*`}`. Este último es un comando híbrido que produce un efecto distinto a lo que está escrito delante y detrás del comando. Veamos algunos ejemplos:

`$\frac{x+y}{y-2}$` $\frac{x+y}{y-2}$ `${232 \over x+y}$` ${232 \over x+y}$

e) ***Signo de integral y de sumatorio.*** Cuando deseamos escribir una integral con LaTeX podemos hacerlo utilizando el comando `$\int$`, que produce el símbolo $\int$.

En el caso de que se trate de una integral definida, los límites inferior y superior de la misma tendrán que ser introducidos como si fueran subíndices y exponentes, respectivamente, como se observa en el siguiente ejemplo:

`$\int_{3}^{2} (x^2+y)dx$` $\int_3^2 (x^2+y)dx$

De igual forma, el signo del sumatorio podemos obtenerlo con la instrucción `\sum`, introduciéndose como exponentes y subíndices los límites superior e inferior de la suma, tal y como se muestra en el siguiente ejemplo:

```
$$\sum_{i=1}^{n}(x^2 + x)$$
```

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

f) Otros elementos usuales de las fórmulas suelen ser las ***letras griegas***, tanto minúsculas como mayúsculas. Éstos son de los símbolos que no funcionan fuera del modo matemático. Para introducir las letras griegas utilizaremos una instrucción que se ajuste al siguiente formato `\letra`, escrito en minúscula para las letras minúsculas y en mayúsculas la primera letra para las letras mayúsculas. Veamos los comandos que originan las letras griegas en el Cuadro 2.1.

`\alpha`	α	`\beta`	β	`\gamma`	γ	`\delta`	δ
`\epsilon`	ϵ	`\varepsilon`	ε	`\zeta`	ζ	`\eta`	η
`\theta`	θ	`\vartheta`	ϑ	`\iota`	ι	`\kappa`	κ
`\lambda`	λ	`\mu`	μ	`\nu`	ν	`\xi`	ξ
`\pi`	π	`\varpi`	ϖ	`\rho`	ρ	`\varrho`	ϱ
`\sigma`	σ	`\varsigma`	ς	`\tau`	τ	`\upsilon`	υ
`\phi`	ϕ	`\varphi`	φ	`\chi`	χ	`\psi`	ψ
`\omega`	ω	`o`	o				
`\Gamma`	Γ	`\Delta`	Δ	`\Theta`	Θ	`\Lambda`	Λ
`\Xi`	Ξ	`\Pi`	Π	`\Sigma`	Σ	`\Upsilon`	Υ
`\Phi`	Φ	`\Psi`	Ψ	`\Omega`	Ω		

Cuadro 2.1: Letras griegas.

Estos símbolos pueden escribirse muy fácilmente si nos valemos de las paletas que poseen los editores de archivos.tex, tales como WinShell y WinEdt, de los que hablaremos más adelante.

Al igual que ocurre con los símbolos anteriores, los denominados ***operadores binarios*** solo funcionan correctamente en modo matemático. Algunos ejemplos de los mismos pueden verse en el Cuadro 2.2.

`+`	$+$	`-`	$-$	`*`	$*$	`\ast`	$*$
`\pm`	$\pm$	`\sqcap`	$\sqcap$	`\otimes`	$\otimes$	`\star`	$\star$
`\mp`	$\mp$	`\sqcup`	$\sqcup$	`\oslash`	$\oslash$	`\oplus`	$\oplus$
`\times`	$\times$	`\vee`	$\vee$	`\odot`	$\odot$	`\ominus`	$\ominus$
`\div`	$\div$	`\wedge`	$\wedge$	`\bigcirc`	$\bigcirc$	`\bullet`	$\bullet$
`\cap`	$\cap$	`\bigtriangleup`	$\bigtriangleup$	`\wr`	$\wr$	`\circ`	$\circ$
`\cup`	$\cup$	`\bigtriangledown`	$\bigtriangledown$	`\diamond`	$\diamond$	`\cdot`	$\cdot$
`\uplus`	$\uplus$	`\triangleleft`	$\triangleleft$	`\dagger`	$\dagger$	`\amalg`	$\amalg$
`\setminus`	$\setminus$	`\triangleright`	$\triangleright$	`\ddagger`	$\ddagger$		

Cuadro 2.2: Operadores binarios.

La lista anterior la completan los denominados ***símbolos de relaciones***. Vemos algunos ejemplos de los mismos en el Cuadro 2.3.

`<`	$<$	`>`	$>$	`=`	$=$	`\approx`	$\approx$
`\leq`	$\leq$	`\geq`	$\geq$	`\neq`	$\neq$	`\parallel`	$\parallel$
`\in`	$\in$	`\notin`	$\notin$	`\ni`	$\ni$	`\cong`	$\cong$
`\perp`	$\perp$	`\equiv`	$\equiv$	`\sim`	$\sim$	`\simeq`	$\simeq$
`\subset`	$\subset$	`\supset`	$\supset$	`\subseteq`	$\subseteq$	`\supseteq`	$\supseteq$

Cuadro 2.3: Símbolos de relaciones.

Capítulo 3

Caracteres especiales. Tablas y cajas. Enumeraciones. Aspecto del texto

3.1. Algunos caracteres especiales

Aquí repasaremos algunos comandos de LaTeX que nos permitirán seguir escribiendo cuestiones habituales de un modo riguroso.

3.1.1. Tildes

TeX permite el uso de acentos de distintos idiomas. Para el castellano utilizaríamos el comando `\'{ }`, poniendo entre las llaves el carácter que queremos acentuar. El comando anterior puede reducirse a `\'` cuando se va a escribir una tilde sobre una letra.

Para colocar el acento sobre una i, primero se debería eliminar el punto que hay sobre la letra. Esto se consigue combinando la instrucción anterior con la instrucción `\i`. Por ejemplo para escribir "*diría*" pondríamos `dir\'{\i}a`. No obstante, versiones recientes de TeX tienen definido el comando `\'{i}` directamente como *í*, si cargamos el paquete `\usepackage[T1]{fontenc}`.

3.1.2. Puntos suspensivos

Si escribimos los *puntos suspensivos* como tres puntos normales quedarían demasiado pegados entre sí: ... Para evitar esto, TeX ofrece unos comandos especiales con los que se pueden hacer distintos tipos de puntos suspensivos:

- `$\ldots$`: Puntos suspensivos en la línea inferior del texto. Ejemplo: $\ldots$
- `$\cdots$`: Puntos suspensivos centrados con respecto al texto. Ejemplo: $\cdots$
- `$\ddots$`: Puntos suspensivos en diagonal. Ejemplo: $\ddots$

- `$\vdots$`: Puntos suspensivos en vertical. Ejemplo: ⋮

3.2. Construir una tabla con TEX

Tanto WinShell como WinEdt ofrecen un *Asistente para tablas*. Aunque no lo usemos, podremos construir directamente las tablas nosotros mismos. En general existen dos entornos muy habituales para construir tablas: uno es más apropiado para tablas de texto y el otro es más apropiado si queremos construir una tabla con fórmulas y símbolos matemáticos.

3.2.1. Contenido de la tabla

Todas las tablas pueden comenzar con la orden `\begin{table}` y finalizar con `\end{table}`, que crea un entorno especial para numerar las tablas.

El siguiente paso consistiría, si se quiere, en centrar la tabla. Para ello, utilizaremos el entorno `center`, con lo que escribiríamos el contenido de la tabla entre los comandos `\begin{center}` y `\end{center}`. Si no utilizamos estos comandos, la tabla quedará, por defecto, alineada a la izquierda de la hoja.

Tras ello solo nos resta introducir la tabla, lo cual haremos con uno de los siguientes entornos: `tabular` o `array`. El entorno `tabular` es el entorno más idóneo para escribir tablas de texto, introduciendo su contenido como sigue:

`\begin{tabular}{`Argumento`}`
Contenido
`\end{tabular}`

Cuando queremos introducir fórmulas y símbolos en la tabla, es más cómodo emplear el entorno `array`. Una tabla generada con el entorno `array` aparecerá escrita completamente en modo matemático sin necesidad de declarar dicho modo en cada una de las celdas. Para insertar una de estas tablas se escribe:

`$$ \begin{array}{`*Argumento*`}`
Tabla
`\end{array} $$`

Tanto el entorno `tabular` como el entorno `array` van obligatoriamente acompañados de un argumento que aporta información sobre el formato de las columnas. Debe haber un argumento por cada columna, así como posibles argumentos extra para los bordes de las columnas y para el espacio entre las mismas. Los símbolos que definen el formato de las columnas son:

- ***l*** : Justifica a la izquierda el contenido de la columna.
- ***c*** : Centra el contenido de la columna.
- ***r*** : Justifica a la derecha el contenido de la columna.
- **|** : Dibuja una línea vertical separando las columnas.

- || : Dibuja dos líneas verticales separando las columnas.

Cuando escribamos el contenido de la tabla, también debemos tener en cuenta el número de filas que va a tener la misma. Cada fila consiste en una secuencia de columnas separadas entre sí por el símbolo &. Para finalizar una fila escribimos \\.

Para dibujar en la tabla líneas horizonales, utilizamos el comando `\hline`. Este puede aparecer antes de la primera fila o inmediatemente después de la terminación de la fila con \\. Este comando dibuja una línea horizontal, que ocupa todo el ancho de la tabla, inmediatamente después de la fila que acaba de terminar o al comienzo de la tabla si el comando va al principio. Si queremos que haya una línea doble escribiremos el comando dos veces: `\hline\hline`.

A veces se necesita rodear la tabla (u otro elemento) por medio de algún delimitador, como (, [, {, |, etc. Para hacer esto, lo más conveniente es utilizar `\left(` (o el delimitador correspondiente en lugar del paréntesis) y `\right)`. En caso de que no se quiera utilizar delimitador en uno de los dos lados, es obligatorio el uso de un punto en lugar del delimitador correspondiente. Un ejemplo de esto es el siguiente sistema de ecuaciones:

```
$$ \left\{
\begin{array}{l}
x+y=1\\
x-y=0
\end{array}
\right. $$
```

$$\left\{ \begin{array}{l} x+y=1 \\ x-y=0 \end{array} \right.$$

Por último, si queremos ponerle un nombre a la tabla, utilizamos el comando `\caption{NombreTabla}`. Este comando lo colocaremos después de `\end{array}$$` o `\end{tabular}`, pero antes de `\end{table}`.

Veamos algunos ejemplos sencillos de tablas o cuadros:

Si escribimos:

```
\begin{table}
\begin{center}
\begin{tabular}{||c|c|c||}
\hline\hline
renta & precio & capital \\
\hline
inter\'es & deuda & beneficios \\
\hline
d\'eficit & excedente & saldo \\
\hline\hline
```

```
\end{tabular}
\caption{Variables econ\'omicas}
\end{center}
\end{table}
```

obtenemos el Cuadro 3.1.

renta	precio	capital
interés	deuda	beneficios
déficit	excedente	saldo

Cuadro 3.1: Variables económicas

Del mismo modo, si escribimos:

```
\begin{table}
\begin{center}
$$\begin{array}{|ccc|}
\hline\hline
\alpha & \beta & \gamma \\
\hline
\delta  & \epsilon  & \varepsilon  \\
\hline
\zeta  & \eta  & \theta  \\
\hline\hline
\end{array}$$
\caption{Letras griegas}
\end{center}
\end{table}
```

el resultado aparece representado en el Cuadro 3.2:

α	β	γ
δ	ϵ	ε
ζ	η	θ

Cuadro 3.2: Letras griegas

3.2.2. Ubicación de la tabla en el documento

Para finalizar esta Sección, comentaremos muy brevemente algunas instrucciones que LaTeX proporciona para la ubicación de las tablas en un documento. En general, todo lo que se incluye en un entorno `table` (y lo mismo ocurrirá con el entorno `figure` que veremos en el Capítulo 4) es tratado como un único elemento flotante sobre el texto.

Por defecto, TeX ubica la tabla donde considera más apropiado. Si queremos forzar una determinada ubicación, debemos utilizar un *designador de colocado*. Éste se coloca al comienzo de la tabla, con la instrucción \begin{table}[*Posición*].

Los principales parámetros para indicar la posición de una tabla (o de cualquier otro elemento flotante) son:

- ***h*** (here): muy próximo al lugar en el texto donde se ha introducido.
- ***t*** (top): en la parte superior de una página.
- ***b*** (bottom): en la parte inferior de una página.
- ***p*** (page): en una página especial que sólo contenga elementos flotantes.

No obstante, no se puede asegurar que TeX sitúe la tabla en el lugar exacto que le hayamos indicado.

3.3. Construir una caja con TeX

Una *caja* es un objeto que es tratado por TeX como un único carácter. Una caja, por tanto, no puede romperse entre dos líneas o dos páginas. Los entornos más sencillos para hacer cajas son \fbox{*Texto*} y \mbox{*Texto*}. El primero escribe el *Texto* que queramos en una caja con un marco, mientras que el segundo hace lo mismo con el *Texto*, pero sin incluir el marco. En ambos casos el ancho de la caja se ajusta automáticamente a la longitud del texto. Veamos cómo funcionan ambos comandos:

Esto está escrito con \fbox

Esto está escrito con \mbox

TeX ofrece otros comandos que son una generalización de los dos anteriores. El comando \framebox[*Ancho*][*Posición*]{*Texto*} es una generalización de \fbox{*Texto*}, mientras que el comando \makebox[*Ancho*][*Posición*]{*Texto*} lo es de \mbox{*Texto*}.

Estos dos comandos permiten al usuario especificar el ancho de la caja y la posición del texto dentro de ella. Para el ancho escribiremos la unidad de medida que queramos emplear como referencia (véase la Sección 3.6 para conocer las principales unidades de medida empleadas en TeX), mientras que para indicar la posición tenemos las siguientes posibilidades:

- ***l***: Justifica a la izquierda el contenido de la caja.
- ***c***: Centra el contenido de la caja.
- ***r***: Justifica a la derecha el contenido de la caja.

Para poner un ejemplo, podemos combinar las cajas con otros entornos, como una tabla, y obtener efectos como marcos que rodean varias cajas. Así si ponemos:

```
\begin{table}
\begin{center}
\begin{tabular}{|c|}
\hline\\
\makebox[8cm][r]{Esto es una caja sin marco}\\
\framebox[80mm][c]{Esto es una caja con marco} \\
\hline
\end{tabular}
\end{center}
\end{table}
```

Estaremos poniendo dos cajas dentro de una tabla, el resultado será el que aparece en el Cuadro 3.3.

Esto es una caja sin marco
Esto es una caja con marco

Cuadro 3.3: Dos cajas dentro de una tabla.

3.4. Enumeraciones. Entornos enumerados

En esta Sección presentaremos dos entornos útiles para construir listas. El entorno `itemize` se emplea para listas sencillas, mientras que el `enumerate` genera relaciones numeradas.

3.4.1. Entorno `itemize`

Las características principales de este entorno son:

- Cada entrada viene precedida de un elemento que en *article* es un punto negro llamado *"bullet"* y en *book* un cuadrado negro.
- El texto de cada entrada es de longitud ilimitada.

- El entorno permite hacer una lista con distintos niveles de esquematización. De hecho, solo admite cuatro niveles. Cada nivel será indicado por un carácter distinto y con una sangría.

Por ejemplo, si escribimos:

```
\begin{itemize}
\item Tema 1: Monopolio.
\begin{itemize}
\item Maximizaci\'on de beneficios.
\item Discriminaci\'on de precios.
\begin{itemize}
\item Discriminaci\'on de primer grado.
\item Discriminaci\'on de segundo grado.
\item Discriminaci\'on de tercer grado.
\end{itemize}
\end{itemize}
\item Tema 2: Oligopolio.
\begin{itemize}
\item Modelo de Bertrand.
\item Modelo de Cournot.
\end{itemize}
\end{itemize}
```

El resultado es:

- Tema 1: Monopolio.
  - Maximización de beneficios.
  - Discriminación de precios.
    - Discriminación de primer grado.
    - Discriminación de segundo grado.
    - Discriminación de tercer grado.
- Tema 2: Oligopolio.
  - Modelo de Bertrand.
  - Modelo de Cournot.

3.4.2. Entorno `enumerate`

Las características principales de este entorno son:

- Las etiquetas son una secuencia ordenada de números.

- Al igual que con el entorno `itemize`, el texto de cada entrada es de longitud ilimitada y es posible hacer una lista con distintos niveles de esquematización.
- La etiqueta del nivel principal en *article* son números arábicos *1, 2 ...*; en el primer sub-nivel se emplean letras minúsculas *(a), (b) ...*; y para el siguiente subnivel se utilizan números romanos en minúsculas *(i), (ii) ....*

Por ejemplo, si escribimos ahora:

```
\begin{enumerate}
\item Tema 1: Monopolio.
\begin{enumerate}
\item Maximizaci\'on de beneficios.
\item Discriminaci\'on de precios.
\begin{enumerate}
\item Discriminaci\'on de primer grado.
\item Discriminaci\'on de segundo grado.
\item Discriminaci\'on de tercer grado.
\end{enumerate}
\end{enumerate}
\item Tema 2: Oligopolio.
\begin{enumerate}
\item Modelo de Bertrand.
\item Modelo de Cournot.
\end{enumerate}
\end{enumerate}
```

el resultado es:

1. Tema 1: Monopolio.
    a) Maximización de beneficios.
    b) Discriminación de precios.
        1) Discriminación de primer grado.
        2) Discriminación de segundo grado.
        3) Discriminación de tercer grado.
2. Tema 2: Oligopolio.
    a) Modelo de Bertrand.
    b) Modelo de Cournot.

Tanto en este entorno como en el entorno `itemize`, TEX nos permite cambiar las etiquetas de los elementos de la lista. Para ello, escribiremos `\item`[*Etiqueta*], poniendo en *Etiqueta* la opción elegida para la presentación de la lista.

Veamos su utilización práctica. Si escribimos:

```
\begin{enumerate}
\item[A] Tema 1: Monopolio.
\begin{enumerate}
\item[A.1] Maximizaci\'on de beneficios.
\item[A.2] Discriminaci\'on de precios.
\begin{enumerate}
\item[A.2.1] Discriminaci\'on de primer grado.
\item[A.2.2] Discriminaci\'on de segundo grado.
\item[A.2.3] Discriminaci\'on de tercer grado.
\end{enumerate}
\end{enumerate}
\item[B] Tema 2: Oligopolio.
\begin{enumerate}
\item[B.1] Modelo de Bertrand.
\item[B.2] Modelo de Cournot.
\end{enumerate}
\end{enumerate}
```

El resultado es:

A Tema 1: Monopolio.

A.1 Maximización de beneficios.

A.2 Discriminación de precios.

A.2.1 Discriminación de primer grado.

A.2.2 Discriminación de segundo grado.

A.2.3 Discriminación de tercer grado.

B Tema 2: Oligopolio.

B.1 Modelo de Bertrand.

B.2 Modelo de Cournot.

3.5. Modificadores del aspecto del texto

En esta Sección veremos distintas herramientas necesarias para definir la apariencia que va a tener el texto final.

3.5.1. Márgenes del documento y diseño de la página

Cada tipo de documento (`article, book, slides, report`, etc.) tiene por defecto una determinada apariencia. Ésta, sin embargo, puede ser alterada a gusto del usuario.

Presentaremos ahora algunos comandos que permiten modificar los márgenes del documento y controlar el aspecto final de la página.

- Los márgenes horizontales del texto pueden fijarse especificando los siguientes comandos:

  `\hoffset`: Delimita el margen izquierdo de impresión.

  `\oddsidemargin`: Fija el margen izquierdo (para las páginas impares).

  `\evensidemargin`: Fija el margen izquierdo (para las páginas pares). Es importante señalar que, salvo que tengamos un documento a doble cara, los márgenes izquierdo y derecho deberían coincidir.

  `\textwidth`: Especifica el ancho de la línea del texto.

- Los principales comandos para controlar los márgenes verticales son:

  `\voffset`: Delimita el margen superior de impresión.

  `\topmargin`: Define la distancia vertical entre el margen superior de impresión y la parte superior del encabezado de la hoja.

  `\headheight`: Define la altura del encabezado.

  `\headsep`: Permite definir la distancia entre la base del encabezado y la parte superior del cuerpo del texto.

  `\textheight`: Especifica la altura de la página.

3.5.2. Tipos y tamaños de letra

Tipos de letra

Los tipos de letra más habituales son:

`\textnormal{normal}`	normal	`\textit{cursiva}`	*cursiva*
`\textbf{negrita}`	**negrita**	`\texttt{máquina}`	`máquina`

Tamaños de letra

Por defecto, los caracteres que escribe LaTeX son de `10pt`, aunque también pueden tomarse como tamaño por defecto los carácteres de `11pt` y `12pt`. Sin embargo, en cualquier entorno podemos alterar el tamaño de los caracteres; los tamaños de letra más habituales son:

\tiny{letra pequeñísima}	letra pequeñísima
\scriptsize{letra muy pequeña}	letra muy pequeña
\footnotesize{letra de tamaño de nota a pie}	letra de tamaño de nota a pie
\small{letra pequeña}	letra pequeña
\normalsize{letra normal}	letra normal
\large{grande}	grande
\Large{más grande}	más grande
\LARGE{muy grande}	muy grande
\huge{enorme}	enorme
\Huge{la más grande}	la más grande

3.5.3. Separaciones de párrafos y diseño

TEX determina automáticamente las separaciones entre palabras y oraciones, el formato de los párrafos, etc. En esta Subsección estudiamos algunos parámetros que permiten alterar esta configuración.

Separaciones horizontales

Para hacer separaciones horizontales entre caracteres se puede utilizar el comando \hspace{*Longitud*}. Cuando queremos que la separación se realice aunque coincida con el final o el principio de una línea, debemos utilizar \hspace*{*Longitud*}. La longitud la definiremos en la unidad de medida que queramos emplear como referencia (*cm, mm*, etc.). Por ejemplo, para hacer un espacio de 2 cm. escribiremos \hspace{2cm} y obtendremos un espacio de 2 cm.

Separaciones verticales

Por defecto, TEX hace un interlineado sencillo entre las líneas de un documento; se admiten separaciones mayores, empleando la orden \linespread{*Interlineado*} en el preámbulo del documento. Se utiliza la orden \linespread{1.3} para documentos

con un interlineado de espacio y medio y \linespread{1.6} para documentos a doble espacio.

TEX permite igualmente forzar separaciones especiales entre dos párrafos con la orden \vspace{*longitud*}. Esta orden se debe indicar siempre entre dos renglones vacíos. Cuando queremos que la separación se realice aunque coincida con el final o el principio de una página, debemos utilizar \vspace*{*Longitud*}.

Al igual que con \hspace, la longitud la definiremos en la unidad de medida que queramos emplear como referencia (*cm, mm*, etc.). Por ejemplo, para hacer un espacio vertical de 0.65 cm. entre dos líneas, escribiremos \vspace{0.65cm}.

Obtendremos un espacio

de 0.65 cm.

Diseño de párrafos

El parámetro \parindent{*Longitud*} también nos permite modificar el formato de los párrafos. Si escribimos la orden \setlength{\parindent}{*Longitud*} en el preámbulo del documento, definiremos la sangría de la primera línea de todos los párrafos del documento.

Si queremos que la orden afecte solo a una determinada parte del documento, escribiremos \parindent{*Longitud*}, antes del primer párrafo cuya sangría queramos alterar.

Por ejemplo, hemos escrito \setlength{\parindent}{3cm} al comienzo de este párrafo para hacer que la primera línea tenga una sangría de 3 cm.

3.6. Principales unidades de longitud

En esta Sección estudiamos las unidades de longitud más utilizadas en TEX. La siguiente lista nos muestra algunas de estas unidades de longitud, así como su equivalencia en el sistema métrico decimal.

mm **(milímetro):** Un milímetro es ||.

cm **(centímetro):** Un centímetro es | |.

in **(pulgada):** Una pulgada (unos 2.54 cm. aproximadamente) es | |.

pt **(punto):** Un punto (aproximadamente 1/3 de milímetro) es ‖.

em: Es aproximadamente el ancho de una letra m en el tipo de letra actual. Aquí, un em es | |.

ex: Es aproximadamente el alto de una letra x en el tipo de letra actual. Aquí, un ex es | |.

Capítulo 4

Figuras en TEX: dibujos y gráficos. Creación de archivos .PS y .PDF

4.1. Dibujos en TEX

En primer lugar, recordemos cómo crear una caja visible o invisible alrededor de un texto. Los comandos \mbox{*Texto*} y \fbox{*Texto*} producen una caja que contiene al *Texto* y cuya altura se ajusta al tamaño de la letra del *Texto*, aunque el segundo comando pone además un marco a dicha caja. Los comandos \makebox[*Ancho*][*Posición*]{*Texto*} y \framebox[*Ancho*][*Posición*]{*Texto*} extienden a los anteriores en el siguiente sentido: crean cajas (sin y con marco, respectivamente) de anchura *Ancho*, de altura el alto del tamaño de la letra del *Texto* y colocan el *Texto* en cierta *Posición* dentro de la caja. El argumento *Ancho* puede ser cualquier longitud, pero un ancho típico es \width, que ajusta el ancho de la caja al ancho del *Texto*. El argumento *Posición* fija la posición que ocupa el *Texto* dentro de la caja. Dicho argumento puede tomar como valores l, r ó c dependiendo de si lo queremos a la izquierda, derecha o centrado, respectivamente.

Nótese que la instrucción \framebox[*Ancho*]{} genera un rectángulo hueco de anchura el *Ancho* que se indica.

El problema de los comandos \mbox, \fbox, \makebox y \framebox es que no controlan el alto de la caja. Esto se subsana con el comando \rule{*Ancho*}{*Alto*}, que genera una caja rellena de negro donde los argumentos *Ancho* y *Alto* indican la anchura y altura de la caja, respectivamente. La instrucción \rule{0cm}{*Alto*} genera una línea vertical invisible cuya altura es la indicada por el argumento *Alto*, con lo cual este comando también puede utilizarse para insertar espacios verticales. Por tanto, si incluimos el comando \rule{0cm}{*Alto*} en el argumento *Texto* del comando \framebox[*Ancho*]{*Texto*} (es decir, si ponemos la instrucción \framebox[ancho]{\rule{0cm}{alto}}) obtenemos un rectángulo hueco de anchura y altura el *Ancho* y *Alto* que se indican.

Por ejemplo, mediante las órdenes:

```
\framebox[5cm]{ }
\rule{2cm}{0.5cm}
\framebox[3cm]{\rule{0cm}{1cm}}
```

obtenemos las siguientes cajas:

Hasta ahora hemos hecho dibujos sin el entorno `picture`, aunque tradicionalmente los dibujos se realizan en dicho entorno. El entorno `picture` es un entorno gráfico para dibujar segmentos, flechas, círculos, cajas y óvalos; además, dentro del dibujo podemos insertar texto en el lugar que deseemos. Dicho entorno necesita de una unidad de medida, que se escribirá en el preámbulo con algo del tipo: `\setlength{\unitlength}{1mm}`.

La unidad de medida a la que hace referencia `\unitlenght` es de `1pt` por defecto. Este comando también podría incluirse en el entorno `picture`, pero resulta más cómodo incluirlo en el preámbulo, ya que en otro caso deberíamos hacerlo en cada uno de los entornos. La instrucción

```
\begin{picture}(x,y)
Textos y gráficos
\end{picture}
```

reserva para nuestro dibujo un área de anchura `x` mm y altura `y` mm. Dentro de un entorno `picture` podemos movernos a cualquier punto de la zona del dibujo, para ello necesitamos establecer un sistema de referencia cartesiano con la unidad de medida que hemos definido en el preámbulo. Cada punto de nuestra zona de dibujo viene representado por unas coordenadas, teniendo en cuenta que la esquina inferior izquierda del área reservada para el dibujo es el origen de coordenadas.

En un entorno `picture` podemos insertar: textos incluyendo fórmulas, cajas sin y con marco, líneas rectas, flechas, círculos y óvalos. Todos estos dibujos se introducen con el comando `\put(a,b){`*Objeto*`}`, donde `(a,b)` son las coordenadas del punto donde se situará la esquina inferior izquierda del *Objeto*.

4.1.1. Cajas

Dentro del entorno `picture` podemos trabajar con las cajas que conocemos (sin y con marco), aunque la sintaxis es diferente. Si queremos situar una caja sin y con marco, respectivamente, debemos utilizar los siguientes comandos:

```
\put(a,b){\makebox(Ancho,Alto)[Posición]{Texto}}
\put(a,b){\framebox(Ancho,Alto)[Posición]{Texto}}
```

donde la esquina inferior izquierda de la caja se sitúa en el punto `(a,b)` y el argumento *Posición* consta de dos letras. La primera letra del argumento *Posición* representa

la ubicación del *Texto* horizontalmente y se elige de entre `l` (izquierda), `r` (derecha) y `c` (centrado); de la misma forma, la segunda letra corresponde a la posición del *Texto* verticalmente y se escoge de entre `t` (arriba), `b` (abajo) y `c` (centrado).

Dentro del entorno `picture` también podemos dibujar, por ejemplo, una caja con la línea del marco discontinua. Para ello, modificaremos el comando anterior de la siguiente forma:

\put(a,b){\dashbox{*Longitud*}(*Ancho*,*Alto*)[*Posición*]{*Texto*}}

donde el argumento *Longitud* indica la longitud en mm. de los segmentos que forman la línea discontinua.

4.1.2. Segmentos y vectores

Dentro del entorno `picture`, el comando

\put(a,b){\line(u,v){*Longitud*}}

genera un segmento donde `(a,b)` son las coordenadas del punto de partida, `(u,v)` son las componentes del vector director y el argumento *Longitud* representa la longitud del segmento en el sentido descrito a continuación. El punto final del segmento tiene abscisa `a+`*Longitud*, excepto si el segmento es vertical (`u=0`), en cuyo caso el punto final del segmento tiene ordenada `b+`*Longitud*.

El comando `\line` tiene una fuerte restricción, ya que `u` y `v` tienen que ser números enteros primos entre sí comprendidos entre -6 y 6.

La sintaxis para dibujar una flecha o un vector es:

\put(a,b){\vector(u,v){*Longitud*}}

donde tenemos las mismas consideraciones que en `\line`.

El comando `\vector` también tiene una restricción importante, puesto que `u` y `v` tienen que ser números enteros primos entre sí comprendidos entre -4 y 4.

4.1.3. Circunferencias y círculos

El entorno `picture` proporciona dos comandos para dibujar una circunferencia o un círculo, respectivamente:

\put(a,b){\circle{*Diámetro*}}

\put(a,b){\circle*{*Diámetro*}}

donde `(a,b)` son las coordenadas del centro y el argumento *Diámetro* corresponde al diámetro de la circunferencia.

El comando `\circle` tiene una restricción sobre el argumento *Diámetro*, que depende de la versión de LaTeX instalada en el ordenador.

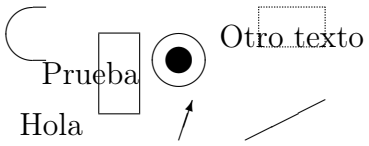
4.1.4. Cajas redondeadas

En primer lugar, observamos que LaTeX entiende por óvalo una caja con las esquinas redondeadas. Para dibujar en el entorno `picture` una caja con marco con las esquinas redondeadas necesitamos el comando:

\put(a,b){\oval(*Ancho*,*Alto*)[*Parte*]}

donde `(a,b)` es el punto central de la caja y el argumento *Parte* indica la parte o partes de la caja redondeada que se desea pintar y consta de una o dos letras de entre la siguientes: `l` (izquierda), `r` (derecha), `t` (superior) ó `b` (inferior).

Veamos con un ejemplo (un poco caótico) cómo el entorno `picture` genera los dibujos:



a partir de las siguientes instrucciones:

```
\begin{picture}(120,60)
\put(0,0){\makebox(20,10)[lt]{Hola}}
\put(30,10){\framebox(15,30)[rc]{Prueba}}
\put(90,35){\dashbox{0.5}(25,15)[cb]{Otro texto}}
\put(85,0){\line(2,1){30}}
\put(60,0){\vector(1,3){5}}
\put(60,30){\circle{20}}
\put(60,30){\circle*{10}}
\put(10,40){\oval(30,20)[l]}
\end{picture}
```

4.2. Otros gráficos en TeX

Para incluir gráficos en los documentos LaTeX necesitamos el paquete `graphicx` (u otro similar), que tenemos que declarar en el preámbulo con una línea como: `\usepackage{graphicx}`. El comando principal del paquete `graphicx` es el siguiente (que mostramos aplicado tanto a un archivo .BMP o a un archivo .EPS):

\includegraphics[width=*Ancho*,height=*Alto*,angle=*Ángulo*]{*grafico.bmp*}
\includegraphics[width=*Ancho*,height=*Alto*,angle=*Ángulo*]{*grafico.eps*}

donde el argumento *Ángulo* indica el ángulo de rotación en grados sexagesimales.

Para incorporar gráficos *Encapsulated PostScript* (.EPS) también existe el paquete `epsfig`, que habría que declarar en el preámbulo con `\usepackage{epsfig}`. La sintaxis del comando principal del paquete `epsfig` es:

`\epsfig{file=`*grafico.eps*`,width=`*Ancho*`,height=`*Alto*`}`

Tanto en el comando `\includegraphics` como en el `\epsfig`, los argumentos *Ancho* y *Alto* especifican las dimensiones que adquirirá el gráfico. Sin embargo, algunos archivos .EPS incorporan las dimensiones de los gráficos, de modo que no es necesario especificar estos dos argumentos; si solo indicamos el argumento *Alto*, el argumento *Ancho* se calcula internamente de forma proporcional (para que no se deforme el gráfico).

En realidad, LaTeX no reconoce los gráficos, sino que solo guarda el espacio para ellos. Para poder visualizar los gráficos en el archivo .DVI, es necesario que los archivos de los gráficos estén en la misma carpeta que el archivo .DVI, ya que éste hace una llamada a estos archivos. El archivo .DVI no incorpora los gráficos, aunque los archivos .PS y .PDF sí que los incluyen. El problema de que a veces no visualicemos los gráficos correctamente está en los drivers, que sí dependen de los periféricos y de los programas que pasan a .PS y .PDF.

El entorno `figure` permite la inclusión de figuras, su localización y numeración y su sintaxis es la siguiente:

`\begin{figure}[`*Posición*`]`
Figura
`\caption{`*Leyenda*`} \label{`*Etiqueta*`}`
`\end{figure}`

donde el parámetro *Posición* indica la posición en que se prefiere ubicar la figura. Puede incluir uno solo de los valores siguientes: `h` (aquí), `t` (al comienzo de una página de texto) ó `b` (al final de una página de texto). El comando `\caption` sirve para poner una leyenda descriptiva y la orden `\label` permite etiquetar la figura, según veremos más adelante.

El entorno `table`, que es análogo al entorno `figure`, vimos que permite la ubicación de tablas o cuadros y su sintaxis era la siguiente:

`\begin{table}[`*Posición*`]`
Tabla
`\caption{`*Leyenda*`} \label{`*Etiqueta*`}`
`\end{table}`

donde el argumento *Posición* y las instruciones `\caption` y `\label` tienen la misma función que en el entorno `figure`.

Si queremos centrar una figura o una tabla, dentro de los entornos `figure` y `table` debemos incluir el entorno `center` de la siguiente forma:

```
\begin{center}
```
Figura o Tabla
```
\end{center}
```

Otra alternativa para centrar consiste en utilizar el comando `\centering`.

Veamos ahora algunos ejemplos del entorno `figure`:

```
\begin{figure}[h]
\begin{center}
\includegraphics[width=40mm,height=30mm,angle=180]{L3.eps}
\end{center}
\caption{Esto est\'a incluido usando el paquete {\tt graphicx}.}
\label{figura1}
\end{figure}

\begin{figure}[h]
\centering\epsfig{file=L3.eps,width=80pt,height=100pt}
\caption{Esto est\'a incluido usando el paquete {\tt epsfig}.}
\label{figura2}
\end{figure}
```

que producen las Figuras 4.1 y 4.2, respectivamente.

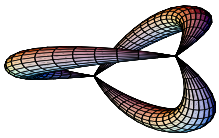


Figura 4.1: Esto está incluido usando el paquete `graphicx`.

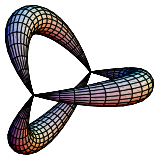


Figura 4.2: Esto está incluido usando el paquete `epsfig`.

4.3. Colores

Como hemos comentado, LaTeX ignora los gráficos (solo les reserva su espacio). Algo parecido ocurre con los colores; también dependen de los *drivers*. Un *driver* es un programa que toma su entrada de un archivo .DVI y produce otro archivo que puede enviarse a una impresora o a la pantalla. Los conversores son específicos del periférico y los comandos que incluyen gráficos o colores son "special", que dependen de los drivers. Por eso, los comandos para incluir gráficos o colores pueden ser distintos para cada PC y para cada sistema operativo. Estos problemas de compatibilidad no impiden que comentemos algo acerca de los colores:

En el preámbulo se debe incluir una línea del tipo:
`\usepackage[dvipdfm,dvipsnames,usenames]{color}`

Se pueden definir colores en uno de los siguientes modelos: *rgb* (proporción de rojo, verde y azul en el color definido), *cmyk* (cian, magenta, amarillo y negro), *gray* (escala de grises) o *named* (68 colores con nombre propio).

Los colores se definen con `\definecolor{`*NombreColor*`}{`*Modelo*`}{`*Números*`}`.

Para utilizarlos, podemos usar, por ejemplo:

`\pagecolor[named]{`*NombreColor*`}` para cambiar el color del fondo.

`\color{`*NombreColor*`}` para cambiar el color del texto en lo sucesivo.

`\normalcolor{`*NombreColor*`}` para volver al color definido por defecto.

`\textcolor[rgb]{0.4,0.3,0.5}{`*Texto*`}` para cambiar el color de *Texto* y utilizar uno con un 4 magnitudes de rojo, por cada 3 de verde y 5 de azul.

`\colorbox{`*NombreColor*`}{`*Texto*`}` para meter *Texto* en una caja coloreada.

`\fcolorbox{`*NombreColor*`}{`*NombreColorMarco*`}{`*Texto*`}` para meter *Texto* en una caja coloreada con marco coloreado.

4.4. Producción de archivos .PS y .PDF

Para obtener un archivo *PostScript*, cuya extensión es .PS, podemos usar el programa DVIPS.exe, que convierte un archivo .DVI en uno .PS. Tanto el programa WinShell como el WinEdt tienen un icono de acceso directo que ejecuta este programa.

Sin embargo, hay varias formas de obtener un archivo del tipo Portable Document File, de extensión .PDF:

- Podemos compilar el archivo .TEX en vez de con LaTeX con PDFLaTeX, con lo que en vez de obtener un archivo .DVI como nos ha ocurrido en los capítulos anteriores, obtendríamos un archivo .PDF directamente.

- Si queremos crear el archivo .PDF desde un archivo .DVI, solo tenemos que usar el programa dvi2pdf.
- En cambio, si queremos obtener el archivo .PDF partiendo del archivo .PS, basta con que usemos uno de los siguientes programas: ps2pdf, distiller o pdfwriter (que viene con GhostScript).

Capítulo 5

Presentaciones con LaTeX: la clase `prosper`

5.1. Ventajas de utilizar LaTeX en presentaciones

Crear transparencias con LaTeX tiene varias ventajas sobre otros programas que habitualmente se emplean para hacer transparencias. La principal es aprovechar documentos que hayamos escrito anteriormente en LaTeX.

También es muy aconsejable cuando se van a usar muchas fórmulas matemáticas, ya que su escritura es mucho más rápida que en los procesadores de texto habituales y su almacenamiento mucho menos voluminoso.

Si con esto lo anteriormente comentado aún no hemos convencido a la gran mayoría, podemos aún dar un argumento más en favor de la utilización de LaTeX para la elaboración de presentaciones: LaTeX permite hacer exactamente todo lo que se nos pueda ocurrir para realizar la presentación, utilizando exclusivamente programas de libre distribución y además es relativamente sencillo generar archivos .PDF, que pueden visualizarse en cualquier ordenador.

5.2. Conocimientos básicos sobre la clase `prosper`

La clase `prosper` permite realizar diapositivas de alta calidad. La mayoría de archivos que editemos en esta clase serán luego traducidos a dos formatos distintos:

Formato PostScript (PS): para imprimir las diapositivas una vez creadas.

Formato Portable Document Format (PDF): para las presentaciones que vayan a ser mostradas en ordenador con Acrobat Reader en modo de pantalla completa.

Existe un formato intermedio, consistente en los archivos .DVI, que no sirve para imprimir o visualizar la presentación, sino que es un paso necesario en el proceso. El esquema general para utilizar `prosper` se indica a continuación.

5.2.1. Pasos para crear una presentación

1. Crear el documento, según las indicaciones que haremos después.

2. Compilar el archivo .TEX, obteniendo de este modo el correspondiente archivo .DVI.

3. Pasar el archivo .DVI a formato .PS. Si solo queremos imprimir las diapositivas, no hace falta realizar el siguiente paso.

4. Por último, convertiremos el archivo .PS en uno .PDF. Normalmente, estos pasos se realizan mediante la selección de iconos del procesador de texto que ya está preparado para escribir en LaTeX, pero también es posible ejecutar órdenes como: `dvips -P file.pdf -GO file.dvi -o`

Cuando las diapositivas creadas con `prosper` se traducen a archivos .PDF se obtiene una ventaja adicional: pueden añadirse efectos de animación en la transición entre diapositivas. Las distintas opciones para mostrar la nueva diapositiva son:

Split: Dos líneas horizontales barren la pantalla desde los extremos.

Blinds: Múltiples líneas horizontales van mostrando la nueva diapositiva.

Box: Una caja va creciendo desde el centro y revela los nuevos contenidos.

Wipe: Una línea vertical barre la pantalla hacia la derecha, como una cortina.

Dissolve: La nueva imagen surge en cuadraditos que se distribuyen por la pantalla de forma aleatoria.

Glitter: Combinación de los dos efectos anteriores.

Replace: Simplemente sustituye una diapositiva por la siguiente.

5.2.2. Estructura de una presentación

Cada diapositiva se inicia con `\begin{slide}` y termina con `\end{slide}`. La estructura general del preámbulo es:

\documentclass[*Opciones*]{prosper}

\title{*Nombre*}
\author{*Autores*}

Ya dentro del cuerpo del documento, especificaremos el contenido de cada diapositiva de la forma:

\begin{slide}[*Transición*]{*Diapositiva 1*}
  Contenido 1
\end{slide}
⋮
\begin{slide}[*Transición*]{*Diapositiva n*}
  Contenido n
\end{slide}

donde *Transición* corresponde al modo en que se le indica a `prosper` que realice el paso de una transparencia a otra; *Diapositiva i* es el título que se le da a la dipositiva i-ésima; y *Contenido* es lo que se quiere que aparezca en la diapositiva en cuestión.

Además de `\title` y `\author`, que sirven para escribir el título y los autores de la presentación, hay otros comandos útiles que pueden ser incorporados en el preámbulo. A continuación indicamos algunos de ellos:

`\subtitle`: Escribe un subtítulo para la presentación.

`\email`: Escribe los correos electrónicos de los autores.

`\institution`: Escribe la institución o empresa a la que pertenecen los autores.

`\slideCaption{`*Leyenda*`}`: Pone la *Leyenda* al fondo de cada transparencia. Por defecto, aparece el título de la presentación.

`\displayVersion`: En vez de escribir la leyenda que defina el usuario para el modo final, utiliza los datos que se escriben en el modo borrador.

`\DefaultTransition{`*Trans*`}`: Indica el tipo de transición entre diapositivas que se va a definir por defecto. Si no se indica nada, `prosper` emplea el modo `Replace`.

5.2.3. Opciones

La clase `prosper` tiene distintas opciones posibles a utilizar. A continuación mostraremos algunas distintas a las establecidas por defecto (cuya descripción indicamos entre paréntesis):

`draft`: El archivo se compila en modo borrador y las figuras se sustituyen por cuadros. Al final de todas las diapositivas se muestra la fecha y hora de compilación junto con el nombre del archivo (la opción por defecto, `final`, hace que se compile el archivo en modo definitivo, colocando las figuras en su lugar y mostrando en cada diapositiva el texto definido, opcionalmente, mediante el comando `\slideCaption`, a excepción de que aparezca el macro `\displayVersion` en el preámbulo, el cual haría aparecer lo mismo que en el modo borrador).

`slideColor`: Las diapositivas usarán muchos colores. Hay que tener cuidado con el efecto cuando se va a imprimir en blanco y negro (`slideBW`, la opción activa por defecto, emplea una cantidad limitada de colores y se recomienda para imprimir la presentación en blanco y negro).

`nototal`: Al final de cada diapositiva se muestra solo el número de la diapositiva actual (La opción por defecto, `total`, hace que también aparezca el número total de diapositivas).

`colorBG`: El color de fondo de la diapositiva dependerá del estilo elegido (si no se especifica esta opción, la que viene por defecto, `nocolorBG`, hace que el fondo sea siempre blanco).

`pdf`: El archivo se compila para producir uno en formato .PDF que podrá emplearse en presentaciones con vídeo-proyector (por defecto está activa la opción `ps` que produce un archivo .PS para su impresión).

`accumulate`: Los macros `\onlySlide`, `\untilSlide` y `\fromSlide`, de los que trataremos luego, interpretan sus argumentos en modo `ps` (por defecto, la opción activa es `noaccumulate` que indica que no se interpreten esos argumentos en modo `ps`). Es posible modificar la opción localmente mediante el uso de los macros `\Accumulatetrue` y `\Accumulatefalse`.

5.3. Macros que pueden aparecer en el entorno `slide`

`\FontTitle{C}{BN}`: Cambia la fuente y/o color de los títulos de las diapositivas.

`\FontText{C}{BN}`: Cambia la fuente y/o color del texto de la diapositiva.

Para los dos macros anteriores, el primer argumento es para las diapositivas en color y el segundo para las que son en blanco y negro.

`\fontTitle{xx}`: Escribe su argumento utilizando la fuente y el color del título.

`\fontText{xx}`: Escribe su argumento utilizando la fuente y el color del texto.

`\ColorFoot{`*Color*`}`: El pie de página se escribirá con el color *Color*.

`\PDFtransition{`*Trans*`}`: Usa *Trans* como efecto de transición entre la diapositiva anterior y la actual.

`\myitem{`*Nivel*`}{`*Def*`}`: Define el objeto del nivel *Nivel* (donde *Nivel* puede tomar el valor 1, 2 ó 3) como *Def*. Por defecto, *Def* es un rombo verde para todos los niveles.

5.4. Overlays

Es la forma en la que se animan las diapositivas en modo PDF. Los comandos que explicaremos a continuación pueden usarse para hacer aparecer o desaparecer elementos de una diapositiva. Es necesario incluir el entorno `slide` dentro de un macro `\overlays` de la siguiente forma:

```
\overlays{n}{
\begin{slide}{...}
...
\end{slide}}
```

El argumento `n` del macro `\overlays` indica el número de pasos del que se compone la animación. Los macros mostrados a continuación permiten controlar lo que aparece en cada paso o diapositiva que forma parte de un *overlays*:

`\fromSlide{p}{`*Contenido*`}`: Pone *Contenido* desde la diapositiva p hasta la n.

`\onlySlide{p}{`*Contenido*`}`: Pone *Contenido* en la diapositiva p.

`\untilSlide{p}{`*Contenido*`}`: Pone *Contenido* desde la diapositiva 1 hasta la p.

Si se escribe la primera letra del macro en mayúsculas, ya no será necesario el segundo argumento y se incluirá todo lo que aparezca tras el macro desde esa diapositiva, en esa diapositiva o hasta esa diapositiva, respectivamente.

Debe tenerse en cuenta que los macros que hemos indicado en esta sección solo funcionan correctamente en modo PDF. Si los utilizamos en modo PS, puede que no hagan nada o que hagan otra cosa distinta de la que esperamos.

Cuando lo que queremos es que un contenido reemplace a otro, es necesario definir cajas de dimensión cero. El código más simple para realizar esto sería, por ejemplo:

```
\onlySlide*{1}{\includegraphics}{ejemplo1.eps}}%
\onlySlide*{2}{\includegraphics}{ejemplo2.eps}}%
\onlySlide*{3}{\includegraphics}{ejemplo3.eps}}%
```

El resultado sería que en la primera diapositiva tendríamos la figura `ejemplo1.eps`, reemplazada por `ejemplo2.eps` en la segunda dipositiva y por `ejemplo3.eps` en la tercera. El símbolo de comentario, %, al final de la línea hace que LaTeX no inserte ningún espacio extra por el cambio de línea entre las figuras.

Los macros especificados a continuación permiten elegir el contenido en función de que hayamos escogido el modo `ps` o `pdf`:

`\PDForPS{sipdf}{sips}`: Interpreta el contenido `sipdf` si el modo elegido es `pdf` o `sips` si es `ps`.

`\onlyInPS{`*Contenido*`}`: Interpreta *Contenido* solo si el modo es `ps`.

`\onlyInPDF{`*Contenido*`}`: Interpreta *Contenido* solo si el modo es `pdf`.

5.5. Más información sobre la clase `prosper`

En `prosper` existen varios estilos predefinidos de diapositiva, aunque no todos disponen del mismo espacio para el texto. También existe la posibilidad de definir nuevos estilos.

La forma más cómoda y fácil de profundizar en esta clase es a través de Internet. Para obtener información y bajar los archivos que permite utilizar la clase `prosper`, puede consultarse [8] y para obtener un manual más completo puede verse el realizado por Frédéric Goualard [7].

Capítulo 6

Otros entornos. Fórmulas matemáticas y símbolos especiales. Tablas y cajas

6.1. Tipos de documento: estilos

Para escribir un documento en LaTeX ya dijimos en el Capítulo 1 que es necesario disponer de un editor de textos que admita archivos de tipo ASCII. No es necesario disponer de un editor concreto; se puede emplear aquél que mejor conozcamos, pero no es recomendable utilizar procesadores de textos que utilicen un lenguaje propio, distinto del ASCII.

Para producir un documento, existen un total de 128 caracteres, contando las 26 letras del abecedario doblemente (ya que LaTeX distingue entre mayúsculas y minúsculas), los 10 dígitos, los operadores y demás símbolos representados por solo 7 bits (puede consultarse el mapa de caracteres de ASCII). Así, al procesar un documento en el que aparecen palabras escritas con tilde o una letra tan propia de nuestro idioma como la ñ, nos damos cuenta de que no aparecen tildes ni eñes entre los 128 caracteres. Ello se debe a que en Inglés no se utilizan y, por tanto, la configuración predeterminada de LaTeX no las considera.

Sin embargo, es muy fácil evitar este problema, según se ha visto: bastaría con escribir `\'` para las tildes y `\~n` en el caso de la eñe (podemos buscar la tilde ~ en el mapa de caracteres, dentro de las herramientas del sistema).

No obstante, podemos hacer uso de diferentes paquetes que almacenan un conjunto adicional de comandos y símbolos.

Para poder utilizar o "activar" cada uno de estos paquetes es necesaria la orden `\usepackage{`*NombrePaquete*`}`.

En este caso, bastaría con añadir en el preámbulo del archivo fuente, la instrucción `\usepackage[latin1]{inputenc}` para que se "cargue" en memoria el paquete de símbolos especiales `latin1`. Siempre que se escriban documentos en Español resulta necesario, además, poner la instrucción `\usepackage[spanish]{babel}` que

traduce ciertos "títulos" al castellano, aparte de tener en cuenta algunas peculiaridades tipográficas y activar los patrones de partición de palabras en Castellano; es muy útil cuando queremos escribir un índice, capítulos, bibliografías, etc.

¿Cuándo escribir con LaTeX en Español?

En principio, aunque es más cómodo para los hispanohablantes, esta opción es desaconsejable si vamos a mandar el documento al extranjero o a alguien con otra plataforma diferente a la nuestra.

Escribir en varias columnas:

Por defecto, LaTeX escribe nuestro documento en una sola columna. Sin embargo, podemos cambiar esta circunstancia con las opciones `onecolumn|twocolumn` que componen el texto a una o dos columnas, respectivamente:

`\twocolumns`**:** Permite escribir el documento en 2 columnas. Si en un documento escribimos este comando, a partir de ese momento se abre una nueva página y se empieza a escribir en dos columnas.

El paquete `multicol`**:** El paquete `multicol` permite escribir textos en una o varias columnas (hasta diez) dentro de una misma página, equilibrando la longitud de las columnas para conseguir un efecto estético agradable.

El paquete implementa el entorno `multicols` cuyo uso, muy sencillo, es el que sigue:

`\begin{multicols}{`*Número*`}`
Texto
`\end{multicols}`
donde *Número*: es el número de columnas en que se imprimirá el texto.

`\onecolumn`**:** Abre una nueva página y escribe en una sola columna.

6.2. Partes de un documento: más entornos

En documentos matemáticos o científicos es frecuente encontrarse con estructuras del tipo: axioma, definición, teorema, proposición, etc. En LaTeX, estos entornos se definen fácilmente mediante la orden`\newtheorem` y definiendo los entornos que necesitemos siguiendo la estructura `\newtheorem{`*NombreTeorema*`}{`*EtiquetaTeor*`}`, donde *NombreTeorema* es la palabra que identifica al entorno dentro del archivo fuente y *EtiquetaTeor* el rótulo que aparecerá en el documento. Por ejemplo, puede definirse el entorno `axi` con la orden `\newtheorem{axi}{\sc Axioma}`.

Es importante señalar que todas estas definiciones de nuevos entornos deben colocarse en el preámbulo del archivo fuente. No obstante, para generarlos en el documento es necesario seguir las reglas siguientes:

```
\begin{Nombre}[EtiquetaAdicional]
Texto
\end{Nombre}
```

Pongamos un ejemplo. Si escribimos el entorno siguiente:

```
\begin{axi}[del supremo]
Todo conjunto no vac\'{\i}o de $\mathbb{R}$ que est\'{e} acotado
su\-pe\-rior\-men\-te admite supremo.
\end{axi}
```

AXIOMA 1 (DEL SUPREMO) *Todo conjunto no vacío de* $\mathbb{R}$ *que esté acotado superiormente admite supremo.*

Por defecto, cada uno de los entornos tipo teorema tiene un contador que se inicializa desde cero y que no tienen relación alguna con el Capítulo o la Sección.

Habitualmente, se quiere que los números de los entornos tipos teorema estén conectados entre sí y con el Capítulo y Sección en los que aparece. Para ello, debemos modificar ligeramente la definición que hemos dado para estos entornos.

```
\newtheorem{TEOR}{Teorema}[section]
\newtheorem{Cor}[TEOR]{Corolario}
```

En las dos ordenes anteriores, le estamos diciendo a LaTeX que que para numerar los teoremas considere el número de la sección, de tal modo que al comenzar una nueva sección, también comienze el contador de los teoremas. Además, le indicamos que queremos que los corolarios sigan la misma numeración que los teoremas, y que consideren, por tanto, también la sección en la que están.

Otros ejemplos de entornos enumerados son:

`\newtheorem{defin}{Definición}[chapter]` (Esto solo valdrá en la clase `book`)

```
\newtheorem{axi}{\sc Axioma}
\newtheorem{postul}[axi]{Postulado}
\newtheorem{teorema}{\bf Teorema}[section]
\newtheorem{lem}{\bf Lema}[section]
\newtheorem{definición}{\bf Definición}[section]
\newtheorem{propo}{\bf Proposición}[section]
\newtheorem{corolario}{\bf Corolario}[section]
\newtheorem{ejemplo}{\bf Ejemplo}[section]
\newtheorem{algor}{\bf Algoritmo}[section]
```

6.3. Fórmulas matemáticas más complejas

Conviene advertir que hay muchas formas de producir un mismo efecto con diferentes comandos. Normalmente, explicamos los más sencillos.

6.3.1. Tipos de letras en fórmulas matemáticas

Para utilizar el `tipo de letra de una máquina de escribir`, emplearemos la orden `\texttt{`texto`}` o bien `{\tt` texto`}`. Para escribir en **negrita** usaremos la orden `\textbf{`texto`}` o `{\bf` texto`}`.

Sin embargo, si queremos utilizar otro tipo de letras usadas en textos científicos, debemos introducir otros comandos. En la Sección 2.3 del Capítulo 2, ya vimos como podíamos introducir los carácteres de las letras griegas en modo matemático. En modo matemático también podemos hacer las letras caligráficas (que solo funcionan con las mayúsculas). Para ello, debemos utilizar el comando `\mathcal{`*Letra*`}`.

Otro tipo de letras que resultan de interés en documentos de contenido algebraico, por ejemplo, son las letras góticas. El comando para que una letra tenga la tipografía gótica es `\mathfrak{`*Letra*`}`. Para que LaTeX pueda compilar este comando previamente debemos cargar en el preámbulo el paquete `amssymb`.

Los símbolos que se introducen en modo matemático no pueden ponerse en negrita usando los dos comandos indicados al comienzo de la presente Sección. Para que LaTeX escriba en negrita un símbolo dentro de una fórmula se utiliza el comando `\boldsymbol{`*Símbolo*`}`, que requiere haber cargado en el preámbulo el paquete `amsmath` (u otro similar).

Consideremos los siguientes ejemplos de los comandos antes referidos: las órdenes `\boldsymbol{\gamma}`, `\boldsymbol{\mathcal{A}}`, `\boldsymbol{\mathfrak{I}}` producen $\boldsymbol{\gamma}$, $\boldsymbol{\mathcal{A}}$ y $\boldsymbol{\mathfrak{I}}$, respectivamente.

De igual forma que actúa el comando anterior se comporta la instrucción `\pmb`, también incluida en el paquete `amsmath`.

6.3.2. Símbolos encima de otros

Es frecuente en la simbología científica poner alguna descripción encima de ciertos símbolos. La orden `\stackrel` permite apilar un símbolo encima de otro. Escribiríamos: `\stackrel{`*Arriba*`}{`*Abajo*`}`.

Este comando compone el segundo argumento en el estilo en curso, mientras que el primero lo hace en un tamaño inferior.

Ejemplos de lo que podemos hacer con este comando pueden ser, por un lado, escribir vectores con las órdenes `\stackrel{→}{v}`y `\stackrel{→}{f}`, que generan, respectivamente, $\stackrel{\rightarrow}{v}$ y $\stackrel{\rightarrow}{f}$. Una segunda aplicación puede ser el escribir subíndices de un sumatorio, como por ejemplo `(Ax,x)=\sum_{\stackrel{1\le i\le n} {\scriptscriptstyle 1\le j\le n}} a_{ij}x_i\bar{x}_j` cuyo resultado era $(Ax,x)=\sum_{\stackrel{1\le i\le n}{1\le j\le n}} a_{ij}x_i\bar{x}_j$.

6.3.3. Subrayado

Las órdenes `\underline` y `\overline` se usan para subrayar y colocar una línea encima, respectivamente.

Escribiendo `$x+\underline{8+z-9 \,\overline{\bar{z}^4+1}}-z= \zeta$` obtendríamos: $x+\underline{8+z-9\,\overline{\bar{z}^4+1}}-z=\zeta$

6.3.4. Llaves encima y debajo de textos

Si en vez de una raya deseamos abarcar expresiones con llaves horizontales, usaremos `\underbrace` y `\overbrace`.

Ejemplo: `$$\underbrace{\overbrace{\sum_{ n\ge 1}a_n}^{=s_1} + \overbrace {\sum_{n \ge 1}a'_n}^{=s_2}}_{\mbox{\scriptsize \textquestiondown}=\sum_{n\ge1}(a_n+a'_n)?}$$`
Obtendríamos:

$$\underbrace{\overbrace{\sum_{n\ge1} a_n}^{=s_1} + \overbrace{\sum_{n\ge1} a'_n}^{=s_2}}_{¿=\sum_{n\ge1}(a_n+a'_n)?}$$

6.3.5. Paquete `amsfonts`

El paquete `amsfonts` proporciona un conjunto de modelos de letra (tipos de caracteres) que suelen ser utilizados para representar o identificar ciertos conjuntos. Lo cargaríamos en el preámbulo del documento con la orden `\usepackage{amsfonts}`.

Ejemplos:

- Podríamos escribir el conjunto de los números reales de una forma usual con el comando `$\mathbb{R}$`, que produciría $\mathbb{R}$.
- También podríamos escribir letras góticas, muy usadas en Álgebra Conmutativa, al dar nombre a los ideales: con `$\mathfrak{I}$` obtendríamos $\mathfrak{I}$.

6.3.6. Modo "display"

En un documento podemos encontrarnos fórmulas escritas en modo matemático dentro de texto (esto es, entre $ simple). Ejemplo de esto puede ser la integral definida $\int_a^b f(x)dx$, o bien podemos hallar la fórmula en modo matemático extendido (esto es, entre $$), como por ejemplo la integral definida

$$\int_a^b f(x)dx.$$

Como se puede comprobar, el aspecto de la integral cambia según se esté en modo matemático dentro de texto o en modo matemático extendido; es evidente que el tamaño es distinto.

¿Cómo conseguiremos entonces una fórmula expandida dentro de un texto? Para lograr que la expresión en el modo matemático dentro de texto sea igual a la del modo

matemático extendido, tenemos la instrucción `\displaystyle`. Así, obtendríamos: la integral $\displaystyle\int_a^b f(x)dx$ es una integral definida. Ahora, dentro del texto, se observa que el tamaño de la integral es el mismo que en el modo extendido.

6.4. Símbolos especiales

- **Guiones**. Un guión puesto en la forma usual (-) puede resultar corto. Para conseguir guiones más largos se pueden poner hasta tres guiones seguidos, como en el siguiente ejemplo:

  `Su ex-cu\~nado ---que recientemente contrajo matrimonio--- es el nuevo due\~no de la empresa.` El resultado sería: Su ex-cuñado —que recientemente contrajo matrimonio— es el nuevo dueño de la empresa.

  Hay cuatro tipos distintos de guión. Los tres primeros, utilizados en texto, corresponden respectivamente a: segmentación silábica o escritura de palabras compuestas (-), intervalos numéricos (–) y como alternativa a los paréntesis (—). El cuarto corresponde al signo negativo en Matemáticas (−). En TEX son generados respectivamente por: `-, --, --- , $-$.`

  En cuanto a la segmentación silábica, recordemos que LATEX está escrito en Inglés por lo que, en ocasiones, puede cortar mal las palabras al acabar de escribir un renglón. Si queremos que esto no ocurra y deseamos decirle a LATEX dónde debe cortar la palabra, escribiremos *`PrincipioPalabra\-FinalPalabra`* o lo siguiente: `{`*PrincipioPalabra*`}-{`*FinalPalabra*`}`.

  Si lo que queremos es que LATEX corte una palabra al final de una línea por otro lugar que nos interese más que por el que lo hace, podemos emplear el comando `\-` para indicarle que en ese punto de la palabra puede realizarse la fragmentación silábica. Por ejemplo, si escribimos `es\-tupen\-do`, LATEX entenderá que solo puede cortar la palabra tras la sílaba `es` o tras la sílaba `pen`.

- **Comillas**. Las máquinas de escribir utilizaban el mismo signo para abrir y cerrar comillas. En un trabajo profesional se utilizan símbolos diferentes para abrirlas (“) y para cerrarlas (”). Para abrir comillas se puede utilizar la tecla del acento grave del teclado (`), una o dos veces. Para cerrar comillas se utiliza el apóstrofo (') del teclado. No es aconsejable usar en el archivo fuente el signo que utilizan las máquinas de escribir si tenemos activado el paquete `babel`, ya que puede tener efectos inesperados.

- **Dejar espacios**. Si queremos dejar un pequeño espacio, ya sabemos que podemos dejarlo con el teclado o escribir la orden `\,` . Si queremos dejar un espacio mayor podemos escribir varias veces la instrucción anterior o bien podemos

emplear \hspace{*Longitud*}, donde *Longitud* indica la medida del espacio que queremos dejar.

- **Símbolo del euro**. Lo usaremos sólo en modo texto, no en modo matemático. Utilizaremos la instrucción \euro, habiendo activado previamente en el preámbulo del documento el paquete eurosym; el resultado sería: €.

6.5. Producción de símbolos o comandos propios

Podemos generar la fecha de hoy en cualquier idioma con el comando \today, o también generar operadores de tamaño variable con \mathop, cuya sintaxis sería \mathop{*Expresión*}, o \mathop{*Operador*}\limits_a^b (o, sin argumentos que representen los límites, \mathop{*Operador*}\nolimits), e incluso definir algún símbolo "raro". Para esto último haríamos la definición en el preámbulo y luego la llamada dentro del archivo fuente. Algunos ejemplos de símbolos que podemos crear nosotros mismos pueden ser los que vemos a continuación.

En el preámbulo del archivo fuente escribiríamos:

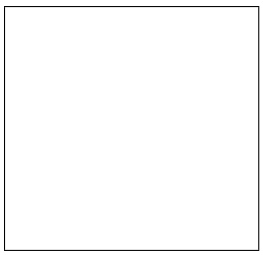
```
\def\C{\,\mbox{\rm l}\!\!\!\mbox{\rm C}}

\def\Bbb#1{{\bf #1}}

\def\N{\mbox{\rm I}\!\mbox{\rm N}}

\def\R{\mbox{\rm I}\!\mbox{\rm R}}

\def\Z{\mbox{\rm Z}\!\!\mbox{\rm Z}}

\newcommand{\CC}{\hbox{\kern 2pt\vrule height6.7pt width0.5pt
 depth-0.2pt \kern -3.2pt {\rm C}}}
```

La llamada de los comandos anteriores se haría en el cuerpo del texto. Escribiendo los comandos $\CC$, $\C$ y $\R$, obtendríamos como resultado $\mathbb{C}$, $\mathbb{C}$ y $\mathbb{R}$.

Recordamos algunas ayudas para definir nuevos comandos:

- **Para dibujar una línea horizontal**: Según lo que se pretenda, puede emplearse tanto el comando \makebox[0.125\linewidth]{\dotfill} como el comando \makebox[0.125\linewidth]{\hrulefill}, donde el primer argumento, que es opcional, es el ancho y el otro, que es obligatorio, es el relleno. \dotfill y \hrulefill son comandos para generar líneas de puntos y líneas continuas, respectivamente.

- **Para hacer un puntal:** \rule[-5mm]{0mm}{11mm}, donde el primer argumento es el desplazamiento vertical respecto de la línea base, el segundo la anchura y el tercero la altura de la caja negra. Todos los argumentos son obligatorios salvo el primero, que es opcional. Veamos un ejemplo: el comando \fbox{\rule{0cm}{3cm} \rule{3cm}{0cm}} da como resultado el siguiente cuadrado:



6.6. Más tablas y cajas

En las Seciones 3.2 y 3.3 del Capítulo 3 ya aprendimos cómo se construían tablas y cajas, pero, ¿qué haríamos para que en una tablas dos celdas estén unidas?

La instrucción \multicolumn permite modificar dentro del entorno `tabular` la estructura inicial de columnas; esto es, pueden incluirse en el entorno `tabular` textos que se extiendan a lo ancho de varias columnas. La estructura del comando es la siguiente:

\multicolumn{*Número*}{*Posición*}{*Texto*}

donde:

Número indica el número de columnas al que afecta el *Texto*.

Posición indica la justificación del texto; debe contener una de las letras `l`, `r` o `c` y también puede contener carácteres como |.

Texto es el contenido de la columna múltiple.

Para ver cómo funciona este comando, habrá que crear antes una tabla:

```
\begin{center}
\begin{tabular}{|c|c|c|}
\multicolumn{2}{c|}{123} & 5 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{tabular}
\end{center}
```

y obtendríamos:

123		5
3	4	5
6	7	8

¿Y si quisiéramos que el texto ocupase más de una línea, esto es, escribir varias líneas dentro de una sola casilla? Entonces podemos actuar de una de las tres formas que indicamos a continuación:

1. Meter una tabla dentro de la propia tabla.

2. Usar el entorno `minipage`, que crea una pequeña página (caja vertical) en la casilla. La sintaxis del entorno es:

   \begin{minipage}[*Posición*]{*Anchura*}
   Objeto
   \end{minipage}

   donde *Posición* es un argumento optativo con el que elegimos la posición que queremos que ocupe el texto dentro de la caja vertical que nos abre el entorno, *Anchura* es un argumento obligatorio que indica la anchura de este entorno y *Objeto* es el contenido de la casilla.

3. Otra orden, cuyo efecto es similar al entorno `minipage`, es \parbox. Al igual que la anterior, construye una caja cuyo contenido se compone en modo párrafo. Su sintaxis es de la forma \parbox[*Posición*]{*Anchura*}{*Objeto*}, donde *Posición, Anchura y Objeto* significan lo mismo que en el entorno `minipage`.

   ¿Podemos usar el entorno `minipage` *o la orden* \parbox *indistintamente?*

   En principio sí, aunque la orden \parbox se usa normalmente para los casos en los que el texto que encierra no sea relativamente extenso y no contenga ninguna declaración de entorno.

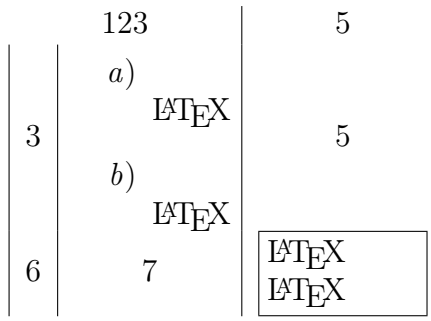
   En diversas situaciones necesitamos, por ejemplo, incluir una lista dentro de nuestra caja. En tales casos, el entorno `minipage` resuelve el problema.

   Aunque estas dos órdenes crean una caja del tamaño que se le indique y con el texto que se quiera en su interior, el marco de la caja no se ve. Si queremos que el marco se vea, usaremos el comando \fbox.

   Ejemplo:

```
\begin{center}
\begin{tabular}{|c|c|c|}
\multicolumn{2}{c|}{123} & 5 \\
3 & \begin{minipage}[c]{2cm} \begin{enumerate} \item \LaTeX \\
\item \LaTeX \end{enumerate} \end{minipage} & 5 \\
6 & 7 & \fbox\parbox[c]{2cm}\LaTeX \\ \LaTeX \\
\end{tabular}
\end{center}
```

   y obtendríamos:



De manera opcional, podemos introducir texto antes o después de la tabla. Cuando construimos una tabla, nos situamos dentro de un entorno `table` en el que introducimos un objeto que será el contenido de la tabla. Si queremos que con el contenido de la tabla aparezca una leyenda indicando el contenido de la tabla, podemos hacerlo con el comando \caption[*LeyendaÍndice*]{*LeyendaTabla*}, que generará una línea en la que aparecerá la palabra Table, Tabla o Cuadro (según la versión de TeX y que esté cargado o no el paquete `babel`) seguido de una numeración similar a la que ya comentamos en la Sección 6.2 de este mismo Capítulo y del texto que hayamos escrito en *LeyendaTabla*, que es un argumento obligatorio. En caso de que hagamos un Índice de Tablas (que veremos en el Capítulo 7) puede sernos de interés que la leyenda que aparezca en dicho índice no sea la misma que con la tabla. Esto podemos arreglarlo con el argumento opcional *LeyendaÍndice* que no es más que el texto que queremos que aparezca en el Índice de Tablas. La orden \caption debe situarse antes o después del contenido del entorno `table`, según queramos que la leyenda aparezca encima o debajo de la tabla, respectivamente.

Si vamos a querer posteriormente referirnos a la tabla que hemos creado en el entorno `table`, deberemos etiquetarla con la orden \label{*Etiqueta*}, donde *Etiqueta* será un cadena alfanumérica que nos servirá para referirnos a ella mediante el comando \ref{*Etiqueta*}, que veremos en la Sección 7.11 del Capítulo 7.

Capítulo 7

Clase de documento `book`

7.1. Elegir la clase de documento `book`

Ya hemos visto distintas clases de documentos, como `report` y `article`. La función principal de la clase de documento `book` se basa en que podemos trabajar con la estructura de un libro. Así, podremos utilizar los entornos `chapter` y `section`, entre otros, que veremos a continuación.

Para elegir la clase `book`, justo al principio del documento escribimos la orden `\documentclass[a4paper]{book}`. En este caso, en el argumento no obligatorio hemos elegido escribir en formato A4.

7.2. Entorno `chapter`

Quizás `chapter` sea el entorno más característico del tipo de documento `book`. Para empezar un capítulo nuevo escribimos `\chapter{`*NombreCapítulo*`}` y se produce lo que podemos ver unas líneas más arriba, en esta misma página.

Este entorno se numera automáticamente, es decir, como ocurre también con `section`, `subsection` o `subsubsection`, no tenemos que numerarlo nosotros. El primer entorno `chapter` será el `Capítulo 1`, el siguiente que escribamos será el `Capítulo 2`, y así sucesivamente.

Puede pasar que en algún momento no nos interese la numeración del capítulo que asigna LaTeX. Así, nos encontramos los casos siguientes:

1. Puede que no queramos que el capítulo esté numerado (que no ponga `Capítulo 1`) pero que mantenga el mismo formato, por ejemplo para la bibliografía, o para los apéndices, etc. En ese caso escribiremos: `\chapter*{`*NombreCapítulo*`}`. Así solo aparecerá el título del capítulo y no lo numerará.

2. Puede que, por alguna razón, no queramos que la numeración del capítulo que vamos a empezar siga con la del capítulo anterior; para este caso escribiremos antes de empezar el nuevo entorno capítulo lo siguiente:

```
\setcounter{chapter}{5}
\chapter{NombreCapítulo}
```

De esta forma, el capítulo que vamos a empezar tendrá la numeración siguiente a la que pongamos en el argumento obligatorio (el que va entre llaves). En el ejemplo, el nuevo capítulo será el 6. Esta forma de modificar el valor de un contador puede ser utilizada con cualquier otro del que se conozca el nombre.

Los contadores son automáticos, es decir, si hemos impuesto que el nuevo capítulo sea el 6, el siguiente a éste será el 7 y así sucesivamente.

Conviene hacer una observación más sobre este tipo de entornos y es que hemos dado un comando para empezar el entorno y concluirá con el comienzo del siguiente entorno del mismo rango.

7.3. Entorno `section`

Las secciones son entornos que van dentro de un entorno capítulo, por tanto, la numeración de éstas se compondrán de dos números: el primero de los cuales hará referencia al capítulo en el que está y el segundo número nos dirá la sección en la que estamos. Para escribir una sección escribimos `\section{`*NombreSección*`}`

Al igual que en el entorno capítulo, si no nos interesa la numeración que nos ofrece LaTeX para la sección tenemos dos posibles casos:

1. Si no queremos que la sección aparezca enumerada, la iniciamos con la línea `\section*{`*NombreSección*`}`.

2. Si no queremos que la sección siga la numeración anterior y queremos imponerle otra, escribiremos:

   ```
   \setcounter{section}{7}
   \section{NombreSección}
   ```

   Así, la sección que empezamos a escribir será la siguiente a la que hemos escrito en el argumento obligatorio. En el ejemplo, sería la 6.8 (no así en este texto).

Si por alguna razón queremos que cada sección empieze en una página diferente, al final de cada una de ellas escribimos: `\newpage` o `\pagebreak` y la siguiente sección empezará en la página que viene a continuación.

7.4. Introducir números de página

En la clase de documento `book`, las páginas aparecen numeradas automáticamente. Si nosotros queremos, por alguna razón, otra numeración distinta de la que LaTeX nos ofrece, debemos cambiarla. Para ello tenemos distintas maneras:

1. Si utilizamos el comando `\addtocounter{page}{3}`, cambiaremos el contador como si de la página anterior a la que estemos en ese momento se produjese un salto de 3 páginas (el argumento obligatorio es el número de páginas y, como tal, va entre llaves).

2. En cambio, si utilizamos el comando `\setcounter{page}{8}`, la página en la que lo escribimos se numerará con el número que hemos puesto en el argumento obligatorio. En nuestro ejemplo, la página en la que se encuentra el comando sería la 8.

Si lo que queremos cambiar es el tipo de número de la página, podemos escribir el siguiente comando: `\pagenumbering{roman}`. Lo que escribimos en el argumento obligatorio es el tipo de número que queremos utilizar para numerar las páginas. Así, en nuestro ejemplo, utilizará los números romanos.

Debemos tener cuidado con este comando, ya que en la página que lo escribamos se reiniciará la numeración, es decir, será la página 1. Así, si no queremos que esto suceda, debemos acompañar este comando con un `setcounter` indicando el número de página en el que queramos comenzar a numerar en romanos.

7.5. Modificadores del aspecto de un texto

7.5.1. Tipos de letra

Podemos cambiar casi todo lo que queramos sobre el tipo de letra. A continuación veremos cómo se modifican un pequeño grupo de letras, pero si queremos que el cambio se haga en todo el documento nos bastará con poner el comando correspondiente en el preámbulo. Así, podemos cambiar su:

1. **Forma**: Las formas fundamentales en las que queremos que aparezcan las letras son:

   a) Para que las letras aparezcan rectas, escribimos `{\upshape` *Texto*`}`. Por ejemplo: Recto.

   b) Para que las letras aparezcan en cursiva, escribimos `{\itshape` *Texto*`}`. Por ejemplo: *Cursiva.*

   c) Para que las letras aparezcan inclinadas, escribimos `{\slshape` *Texto*`}`. Por ejemplo: *Inclinado.*

d) Para que las letras aparezcan en mayúsculas, escribimos `{\scshape` *Texto*`}`. Por ejemplo: MAYÚSCULAS.

e) Para que las letras aparezcan subrayadas, escribimos `\underline{`*Texto*`}`. Por ejemplo: Subrayado.

Si lo que queremos es cambiar la forma de las letras de un entorno entero, nos bastará con escribir los comandos sin las llaves del principio y todo ese entorno en el que aparece el comando, cambiará la forma de su letra. Para tener más controlados los comandos que actúan sobre un entorno completo, suele ser recomendable ponerlos al principio del entorno, con el fin de tenerlos localizados.

f) **Serie**: Las series fundamentales en las que podemos querer que aparezcan las letras son:

1) Para que las letras aparezcan en serie media, escribimos `{\mdseries` *Texto*`}`. Por ejemplo: Media.
2) Para que las letras aparezcan en serie negrita, escribimos `{\bfseries` *Texto*`}`. Por ejemplo: **Negrita**.

También podemos mezclar estos comandos escribiendo unos dentro de otros como, por ejemplo:

```
{\itshape{\bfseries Negrita cursiva}}
```

que nos daría el siguiente resultado: ***Negrita cursiva***

g) **Familia**: Las familias fundamentales en las que podemos querer que aparezcan las letras son:

1) Para que las letras aparezcan en serie Roman, escribimos `{\rmfamily` *Texto*`}`. Por ejemplo: Roman.
2) Para que las letras aparezcan en serie Sans Serif, sirve `{\sffamily` *Texto*`}`. Por ejemplo: Sans Serif.
3) Para que las letras aparezcan en serie Typewriter, vale `{\ttfamily` *Texto*`}`. Por ejemplo: `Typewriter`.

Si hemos cambiado en todo un entorno el tipo de letra y queremos normalizarlo, utilizaremos `\normalfont`.

h) **Tamaño**: Varios tamaños de letra que maneja LaTeX ya se comentaron en el Capítulo 3. El tamaño en el que podemos querer que aparezcan las letras se puede modificar de varias formas:

- Para agrandar las letras dependiendo del tamaño que queramos utilizar escribimos:
  - `\large{`*Texto*`}`: Agranda el tipo de letra un 10 % con respecto al tamaño normal.

- \Large{*Texto*}: Agranda el tipo de letra un 10 % con respecto al tamaño large.
- \LARGE{*Texto*}: Agranda el tipo de letra un 10 % con respecto al tamaño Large.
- \huge{*Texto*}: Agranda el tipo de letra un 10 % con respecto al tamaño LARGE.
- \Huge{*Texto*}: Agranda el tipo de letra un 10 % con respecto al tamaño huge.

- Para empequeñecer las letras dependiendo del tamaño que queramos utilizar:
  - \small{*Texto*}: Empequeñece el tipo de letra un 10 % con respecto al tamaño normal.
  - \footnotesize{*Texto*}: Empequeñece el tipo de letra un 10 % con respecto al tamaño small.
  - \scriptsize{*Texto*}: Empequeñece el tipo de letra un 10 % con respecto al tamaño footnotesize.
  - \tiny{*Texto*}: Empequeñece el tipo de letra un 10 % con respecto al tamaño scripsize.

  Si hemos cambiado en todo un entorno el tamaño de la letra y queremos normalizarlo utilizaremos \normalsize.

7.5.2. Distribuir espacios

Al ir escribiendo un documento puede pasar que en una página quede mucho espacio sobrante (para ello debe ser la última página o haber puesto al final del texto \newpage). LaTeX, por defecto, reparte dicho espacio sobrante uniformemente por el texto, pero nosotros podemos distribuir el espacio a nuestro antojo con los siguientes comandos:

- \vfill: Todo el espacio sobrante (de alto) lo introduce en ese trozo.
- \vfil: La mitad del espacio sobrante (de alto) lo introduce en ese trozo.

Al igual que LaTeX distribuye automáticamente el espacio sobrante de una página también lo hace con el sobrante de una línea. Para distribuir el espacio sobrante en una línea al final de un párrafo, podemos utilizar los dos siguientes comandos:

- \hfill: Todo el espacio sobrante (de ancho) lo introduce en ese trozo.
- \hfil: La mitad del espacio sobrante (de ancho) lo introduce en ese trozo.

En ese mismo sentido, LaTeX corta las palabras y escribe las letras que sobran en la siguiente línea, con el fin de distribuir el espacio de cada línea de un párrafo. También podemos influir en el corte de dichas palabras mediante los siguientes comandos (normalmente escritos en el preámbulo aunque también pueden escribirse en cualquier entorno):

- `\sloppy`: LaTeX intentará cortar el mínimo (prácticamente ninguna) de las palabras del texto. Estira todo lo necesario los espacios entre palabras para cortar el menor número de palabras posibles.

- `\fussy`: LaTeX será muy escrupuloso con los espacios sobrantes en las líneas y, por tanto, cortará bastantes palabras del texto. Esta opción viene dada por defecto, por lo que solo tiene sentido ponerla para anular el comando `\sloppy`. Tanto `\sloppy` como `\fussy` actúan en el documento a partir del lugar en el que aparecen.

7.5.3. Cambiar nombres

Hay algunos entornos a los que LaTeX les asocia un nombre automáticamente. Por ejemplo, al hacer una tabla debajo de ésta nos pondrá, según el tipo de documento `Cuadro 1.1` o `Tabla 1.1` (este entorno viene con numeración). Nosotros podemos querer, por ejemplo, que escriba `Datos 1.1` (para cambiar la numeración de la tabla lo hacemos cambiando el contador, usando `\setcounter`); lo haríamos escribiendo: `\renewcommand\tablename{Datos}`.

Igualmente haremos si lo que queremos cambiar es que al introducir una figura no nos escriba `Figura 1.1` sino `Dibujo 1.1`: `\renewcommand\figurename{Dibujo}`.

Como siempre, si estos comandos los escribimos en el preámbulo afectarán a todo el texto; en cambio, si los escribimos en un entorno solo actuarán en dicho entorno.

7.5.4. Cabeceras, pie de página y comienzo de capítulo

Para diseñar las cabeceras a nuestro gusto tenemos los siguientes comandos fundamentales:

- `\pagestyle{`*Estilo*`}`: Cambiará el estilo de cabecera de todas las páginas del documento.

- `\thispagestyle{`*Estilo*`}`: Cambiará solo el estilo de cabecera de la página en la que hemos escrito este comando.

En el argumento obligatorio de estos comando podemos escribir los siguientes estilos que determinarán el formato de la cabecera:

- `plain`: Tendrá el número de página al pie y centrado.

- `empty`: La cabecera estará vacía.
- `headings`: Tendrá el número de página y el nombre del capítulo en la cabecera.

Si queremos diseñar las cabeceras aún más a nuestro gusto nos harán falta otros comandos para cambiar cabeceras: `\markboth` y `\markright`. Veamos cómo funcionan.

1. `\markboth{`*CabeceraPáginasPares*`}{`*CabeceraPáginasImpares*`}`: A partir de la página en la que hemos escrito este comando y hasta un cambio de sección o capítulo, en la cabecera de las páginas pares escribirá lo que hayamos escrito como *CabeceraPáginasPares* a la zquierda y el número de página a la derecha; y en las impares lo que hayamos escrito como *CabeceraPáginasImpares* a la derecha y el número de página a la izquierda.
2. `\markright{\arabic{chapter}.\arabic{section}.TítuloSección}`: En las páginas impares, escribirá a la derecha de la cabecera el número de página y a la izquierda el nombre que hayamos puesto en la sección siguiendo a los números correspondientes al capítulo (en formato normal) y a la sección (también en formato normal). En las páginas pares saldrá por defecto la palabra `CAPÍTULO`, seguida de su número y el nombre de capítulo a la derecha y el número de página a la izquierda.
3. `\markright{TítuloSección}`: Nos escribirá en las páginas impares a la derecha el número de página y a la izquierda *TítuloSección*. En las páginas pares ocurrirá lo mismo que en el caso anterior.

Otras formas más sofisticadas y que modifican más el diseño del encabezado hacen uso de paquetes que debemos cargar en el preámbulo. A modo de ejemplo, vemos el uso de dos paquetes cuya función es ésa:

1. Para definir el **formato del encabezado y del pie de página**:

   Empezamos escribiendo en el preámbulo:

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

   Una vez operativo el paquete, podemos usar los siguientes comandos:

   `\fancyhead[`*Selectores*`]{`*Cabecera*`}`

   `\fancyfoot[`*Selectores*`]{`*Pie*`}`

   El parámetro obligatorio a introducir en *Selectores* indica el lugar de la cabecera o pie de página donde se pondrá el texto que aparece en la *Cabecera*. Las distintas posibilidades son:

Selectores de página	E	Página par
	O	Página impar
Selectores de campo	L	Parte izquierda
	C	Parte central
	R	Parte derecha

El comando `\fancyhead` corresponde a lo que queremos que aparezca en la cabecera de la página. Por el contrario, `\fancyfoot` sirve para modificar el pie de página.

Veamos varios ejemplos:

`\fancyhead[LO]{\leftmark}`: En las páginas impares, en la parte izquierda del encabezado, aparecerá el nombre de capítulo.

`\fancyhead[RE]{\rightmark}`: En las páginas pares, en la parte derecha del encabezado, aparecerá el nombre de sección.

`\fancyhead[RO,LE]{\thepage}`: Los números de página estarán en las esquinas de los encabezados(a la derecha en las páginas pares y a la izquierda en las impares).

`\renewcommand{\chaptermark}[1]{\markboth{\small \emph{ \thechapter. #1}}}{}`: Cambiará el formato de la cabecera cuando le toque a cada capítulo (tamaño de la letra, etc.) y escribirá el número del capítulo seguido del nombre del mismo.

`\renewcommand{\sectionmark}[1]{\markright{\small \emph{ \thesection.#1}}}`: Cambia el formato de la cabecera cuando le toque a cada sección (tamaño de la letra, etc.) y escribirá el número del capítulo y número de la sección seguido del nombre de la misma.

`\renewcommand{\headrulewidth}{0.6pt}` cambia el ancho de la línea horizontal bajo el encabezado. Si escribimos (además de este comando, o en su lugar) `\renewcommand{\footrulewidth}{0.6pt}`, cambia el ancho de la línea horizontal sobre el pie de página (que en este ejemplo está vacío).

`\setlength{\headheight}{1.5\headheight}` aumenta la altura del encabezado en una vez y media.

`\setlength{\headwidth}{1.5\headwidth}` aumenta la anchura del encabezado en una vez y media.

2. Para definir el **formato del principio de los capítulos**:

   Empezaremos cargando en el preámbulo el paquete `titlesec`, de la siguiente manera: `\usepackage{titlesec}`.

   Podemos empezar, por ejemplo, definiendo una línea gruesa (para luego jugar con ella) de la siguiente manera: `\newcommand{\bigrule}{\titlerule[0.5mm]}`.

Ya sabemos que, por defecto, LaTeX usa el tipo de letra `Huge` en `negrita` para escribir el comienzo de un capítulo. Para cambiar esto usamos (con el paquete anteriormente citado) `\titleformat{\chapter}[display]`, que cambiará el formato de los capítulos dependiendo de los comandos que escribamos en los argumentos. Éstos pueden ser:

- `{\filleft}`: Alineará el texto a la derecha.
- `{\titlerule}`: Hará una línea horizontal.
- `{\bigrule}`: Hará una línea vertical.
- `{\mdseries\small}`: Podemos cambiar el aspecto del texto como queramos.
- `{\chaptertitlename}`: Para cambiar el tamaño de la etiqueta, como `Capítulo` o `Apéndice`. Por ejemplo, para hacerlo tamaño Large escribiríamos `{\Large\chaptertitlename}`
- `{\thechapter}`: Para cambiar el tamaño del número del capítulo. Por ejemplo, para que sea en tamaño Large: `{\Large\thechapter}`
- `[\vspace0.5mm \bigrule]`: después del cuerpo, deja un espacio vertical y traza línea horizontal gruesa.

7.6. Numeración de ecuaciones

Para numerar las ecuaciones hay un entorno específico llamado `equation`; para utilizarlo debemos abrir y cerrar dicho entorno escribiendo dentro la ecuación que queramos numerar (sin escribirla entre dólares). Por ejemplo, si escribimos:

```
\begin{equation}
\sqrt{\int{x^3-2x}}
\end{equation}
```

nos daría la siguiente ecuación numerada:

$$\sqrt{\int x^3 - 2x} \tag{7.1}$$

Si dentro de dicho entorno `equation` quisiéramos escribir algo de texto, tendríamos que hacerlo dentro de una caja. Por ejemplo, en la ecuación anterior sería:

```
\begin{equation}
\sqrt{\int{x^3-2x}}.\mbox{\'Este es el texto que quiero escribir}
\end{equation}
```

$$\sqrt{\int x^3 - 2x}.\text{Éste es el texto que quiero escribir} \tag{7.2}$$

Como se puede observar en ambos ejemplos, LaTeX centrará la ecuación dependiendo del tamaño de la misma.

7.7. Introducir un Índice General

Para que haga un Índice General y ponga el título de éste con el mismo formato que usa con los capítulos, escribimos en el sitio donde queramos colocar el Índice el comando `\tableofcontents`.

Automáticamente hará un índice en ese lugar, incluyendo capítulos y secciones e indicando en las páginas en que se encuentran.

Una forma de incluir o excluir algo del Índice General generado por el programa automáticamente será abrir y modificar el archivo.toc que se genera. En ese archivo.toc se recoje toda la información que hace falta para generar el Índice. Si cambiamos en ese archivo.toc lo que queramos, lo guardamos y compilamos **una sola vez**, el archivo.dvi visualizará todos los cambios que hayamos hecho en el archivo.toc. Hemos de tener cuidado, porque si compilamos una segunda vez, el archivo.dvi volverá a visualizar lo que estaba escrito en el archivo.toc creado por LaTeX. Esto es porque la primera compilación vuelve a modificar el archivo.toc y en la segunda LaTeX usa este archivo modificado.

Como veremos, existen comandos específicos para que estas modificaciones las realice el propio programa.

7.8. Introducir un Índice de Materias

Para poder crear un Índice de Materias (del modo que explicaremos) se necesitan los archivos Findexe.exe, Makeindex.exe y Texindex.exe en \MiKTeX\BIN.

Para hacer un Índice de Materias, en el preámbulo hemos de escribir:

```
\usepackage{makeidx}
\makeindex
```

Si queremos incluir algo que en principio no aparece en el índice, en el texto escribimos: `\index{`*Palabra*`}`

Donde queramos que aparezca el Índice de Materias, escribimos `\printindex` y, automáticamente, en ese lugar se visualizará dicho Índice de Materias.

Para cambiar el nombre de este índice, se puede usar, por ejemplo, la instrucción `\renewcommand\indexname{Glosario}`

7.9. Introducir un Índice de Figuras

Para poder crear un Índice de Figuras, únicamente tenemos que escribir, en el lugar donde queremos que aparezca, el comando `\listoffigures`.

Realizará una lista con todas las figuras del documento, indicándonos la página en la que se encuentran.

7.10. Introducir un Índice de Tablas

Para crear un Índice de Tablas, basta con escribir, en el lugar que queramos que aparezca, el comando `\listoftables`.

Este comando editará una lista con todas las tablas o cuadros del documento, indicándonos la página en la que se encuentran.

7.11. Introducir referencias

Normalmente, las referencias se realizan a entornos enumerados.

En el momento en que queramos hacer referencia a algo que ya está escrito en el texto es cuando usamos una etiqueta. Para hacerlo, nos basta con ir a la parte del texto donde está escrito aquello a lo que queremos hacer referencia y etiquetarlo, escribiendo `\label{`*NombreEtiqueta*`}`.

Luego bastará con escribir (allí donde queremos que salga escrito dónde está a lo que nos referimos) lo siguiente:

- `\ref{`*NombreEtiqueta*`}`: Indicará en qué entorno está, es decir, por defecto el número del capítulo y de sección.
- `\pagref{`*NombreEtiqueta*`}`: Indicará en qué página está.

7.12. Introducir Apéndices.

También podemos querer que un capítulo no se encabece como tal, sino como un anexo o apéndice. Para ello escribimos `\appendix`. Debemos tener cuidado porque a partir de ese comando, todos los demás capítulos que pongamos serán anexos (estos entornos también vienen numerados automáticamente).

7.13. Introducir Bibliografía

La bibliografía en LaTeX es un entorno que necesita un argumento obligatorio de significado variable (siempre vale el 99). Cuando queramos empezarla escribiremos:

```
\bibliographystyle{plain}
\begin{thebibliography}{99}
    Referencias bibliográficas
\end{thebibliography}
```

Las referencias bibliográficas las definiremos de la siguiente manera:

\bibitem{*EtiquetaReferencia*}[*EtiqVisible*] *Referencia bibliográfica*

donde *Referencia bibliográfica* corresponde a la referencia que queremos introducir (con el formato que queramos) y *EtiquetaReferencia* es la etiqueta que le asignaremos a la referencia para citarla a lo largo del documento. *EtiqVisible* es un argumento opcional que será utilizado para señalar el comienzo de la cita en la lista de referencias. Por ejemplo, una referencia bibliográfica podría ser:

```
\bibitem{Ref} Autores, ``Título Libro''. {\it Editorial}, Año.
```

Para referirnos a una cita de un libro, podemos hacer uso de la etiqueta que le hemos asignado y escribimos en el texto \cite{*EtiquetaReferencia*} y LaTeX escribirá el número de referencia. Con \cite{*EtiquetaReferencia*}[*Página*], LaTeX escribirá en el texto, además, el número de página al que se hace referencia.

La bibliografía, los apéndices, los capítulos que no llevaban puesto el nombre, etc., no aparecen en el Índice por defecto. Si queremos que aparezcan, dentro de su entorno escribimos, por ejemplo, en el caso de la Bibliografía:

```
\addcontentsline{toc}{chapter}{Bibliograf\'{\i}a}
```

En el primer argumento obligatorio hemos puesto que escriba en el archivo.toc; en el segundo, hemos asignado rango de capítulo y en el tercer argumento obligatorio hemos indicado cómo queremos que se llame dicho entorno en el Índice.

Capítulo 8

Contadores personalizados. Inserción de gráficos. Traducción de formatos

8.1. Contadores personalizados

A los contadores existentes vistos hasta ahora pueden añadirse otros nuevos (y utilizarse simultáneamente con ellos) creados por nosotros mismos con el comando `\newcounter{`*NombreContador*`}[`*ContadorExistente*`]`. De esta forma, introducimos un contador nuevo de nombre *NombreContador* al cual se le asigna por defecto el valor cero como valor inicial. El contador que definamos no puede llamarse del mismo modo que un contador ya existente puesto que, en tal caso, daría un mensaje de error durante el proceso de compilación.

La definición del contador se puede hacer en cualquiera de las partes del documento y se podrá utilizar a partir de ese momento sin necesidad de redefinirlo de nuevo.

Si en el argumento opcional *ContadorExistente* indicamos un contador que ya existe, entonces el contador *NombreContador* estará subordinado al *ContadorExistente*, con el objeto de que un incremento en `ContadorExistente` provoque que el valor del contador *NombreContador* se reinicie en cero. Como ya hemos visto, esto ya ocurría con el contador de `subsection`, el cual está subordinado al contador de `section`, por lo que siempre que se inicia una nueva sección, el contador de las subsecciones se pone de nuevo a cero.

Una vez introducido el nuevo contador, debemos indicar a partir de qué punto del documento queremos empezar a utilizarlo. Para ello, se utiliza el comando `\usecounter{`*NombreContador*`}`, donde *NombreContador* es el contador que se quiera utilizar.

En el siguiente ejemplo definimos un nuevo contador para enumerar los ejercicios de, por ejemplo, un libro o unos apuntes.

```
\newcounter{ej}
\begin{list}{\bf{Ejercicio \arabic{ej}.}}
```

```
{\usecounter{ej}
\setlength{\labelwidth}{-0.5em}
\setlength{\leftmargin}{-0.1mm}}

\item \'Este va a ser el primer ejercicio...
\item Y \'este el segundo...
...
\end{list}
```

El resultado de este ejemplo es el siguiente:

Ejercicio 1. Éste va a ser el primer ejercicio...

Ejercicio 2. Y éste el segundo...

Obsérvese que se han especificado diferentes opciones para el aspecto de la etiqueta. En primer lugar, tanto la etiqueta *Ejercicio* como el contador definido *ej* aparecerán en negrita, además del estilo de numeración *arabic* para el contador. En segundo lugar, especificamos la anchura de la caja que contiene la etiqueta de la lista con `labelwidth`; y con `leftmargin` se fija el espacio horizontal de sangrado de la lista respecto del entorno anterior.

Hay que comentar también que cada vez que finalicemos la lista y volvamos a empezar una nueva con el mismo contador, éste empezará a enumerar a partir del uno. En el caso de que esto no sea lo que nos interesa, podemos salvar este problema de dos maneras distintas. Una primera sería retrasando la aparición de `\end{list}` hasta el momento que nos interese y la otra solución es, una vez finalizado el entorno `list`, cuando se empiece otra lista, modificar el inicio del contador. Para esto usaremos `setcounter`, indicando entre llaves el número desde el que se desea comenzar. Veamos en el siguiente ejemplo cómo se utiliza:

```
\begin{list}{\bf{Ejercicio \arabic{ej}.}}
{\usecounter{ej}
\setlength{\labelwidth}{-0.5em}
\setlength{\leftmargin}{-0.1mm}}

\item \'Este va a ser el primer ejercicio...

\item Y \'este el segundo...

\end{list}

Comenzamos una nueva lista enumerando los ejercicios
a partir del 5.
```

```
\begin{list}{\bf{Ejercicio \arabic{ej}.}}
{\usecounter{ej}
\setlength{\labelwidth}{-0.5em}
\setlength{\leftmargin}{-0.1mm}}
\setcounter{ej}{5}
\item El primer ejercicio se ha enumerado con 6.

\item El siguiente debe de ser el n\'umero 7.

\end{list}
```

El resultado que obtememos con las líneas anteriores es el siguiente:

Ejercicio 1. Éste va a ser el primer ejercicio...

Ejercicio 2. Y éste el segundo...

Comenzamos una nueva lista enumerando los ejercicios a partir del 5.

Ejercicio 6. El primer ejercicio se ha enumerado con 6.

Ejercicio 7. El siguiente debe de ser el número 7.

8.2. Dibujos en TEX

Habitualmente nos encontraremos con la necesidad de introducir dibujos, gráficas, esquemas, etc. que nos resultarían muy complejos de elaborar con los procedimientos vistos en los capítulos anteriores. Por este motivo se han desarrollado diferentes procesadores gráficos con más precisión, mejor calidad y, sobre todo, mayor compatibilidad con TEX que la que nos aportan las opciones gráficas de, por ejemplo, Word o Microsoft Paint[1].

En primer lugar destacamos el programa MayuraDraw, el cual es de fácil manejo y nos permite exportar el dibujo realizado al formato *.eps* que, como ya comentamos, es el formato más compatible con LATEX. Uno de los inconvenientes de este programa es que es de pago, aunque podemos disponer de una versión de evaluación indefinidamente.

No obstante, siempre es aconsejable tener a mano otros procesadores gráficos ya que puede ocurrir (y os aseguramos que más a menudo de lo que uno desea) que el gráfico desarrollado e insertado en el documento no se vea en la presentación final una vez compilado el documento. Así, también podemos utilizar LATEXCAD y, si el problema persiste, probar a exportar los gráficos en otras extensiones distintas de .EPS como, por ejemplo, al formato Windows Meta File (.WMF).

Algunas direcciones de interés donde encontrar distintos paquetes gráficos con especial énfasis en el color son [4] y [14].

[1] Microsoft Paint ©1981–2004 es marca registrada de Microsoft Corporation.

8.3. Traducción de formatos

En esta Sección vamos a ver cómo crear formatos especiales para la edición de documentos como, por ejemplo, la lista con las calificaciones de los alumnos. Podemos crear un documento con las calificaciones de los alumnos con el formato siguiente:

1 *Alumno 1* ..... Aprobado **(5)**
2 *Alumno 2* ..... Notable **(7)**
3 *Alumno empollón* ..... Sobresaliente **(9'9)**
4 *Super-Paqui* ..... Matrícula de Honor **(10)**

Para ello, hemos escrito las siguientes líneas, que comentamos a continuación:

```
\newcounter{alumno}
\usecounter{alumno}
\newcommand{\Mh}{Matr\'{\i}cula de Honor}
\newcommand{\Sob}{Sobresaliente}
\newcommand{\Not}{Notable}
\newcommand{\Apr}{Aprobado}
\newcommand{\Sus}{Suspenso}
\newcommand{\NP}{No presentado}

\newcommand{\nota}[3]{\par\noindent\stepcounter{alumno}
\makebox[0.9\linewidth]{\arabic{alumno} {\it #1}
\dotfill #2 {\bf (#3)}}}

\nota{Alumno 1}{\Apr}{5} \nota{Alumno 2}{\Not}{7} \nota{Alumno
empoll\'on}{\Sob}{9'9}
```

En primer lugar, hemos creado un contador personalizado de nombre `alumno` (véase la Sección 8.1 para recordar cómo se introducían contadores personalizados). Dicho contador lo utilizaremos para enumerar a los alumnos de nuestra lista. A continuación, por comodidad, definimos seis nuevos comandos con la orden `newcommand` correspondientes a las seis calificaciones distintas que podemos asignar a cada uno de los alumnos.

Una vez definidos los seis comandos para las calificaciones, definimos el comando `nota`, que tendrá tres argumentos obligatorios diferentes. Estos tres argumentos corresponden al nombre del alumno, a la calificación cualitativa y a la calificación cuantitativa, respectivamente. La definición de dicho comando viene dada por una secuencia de órdenes para el estilo de presentación:

1. `\par` salta de línea (en realidad, salta de párrafo) para que aparezca cada calificación en una línea diferente.

2. `\noindent` no sangra la nueva línea que se está comenzando a escribir.

3. `\stepcounter{alumno}` hace avanzar en una unidad el contador `alumno`, además de reiniciar todos los contadores subordinados a `alumno`.

4. `\makebox[0.9\linewidth]` hace que el comando `nota` esté formado por una caja de ancho igual al 90 % del ancho normal de una línea.

5. `\arabic{alumno} {\it #1} \dotfill #2 {\bf (#3)}` son los elementos incluidos en la caja anteriormente definida. En primer lugar, el contador `alumno` en el estilo `arabic`; en segundo lugar, el primero de los argumentos del comando `nota` en itálica; a continuación, se rellena el espacio sobrante con una línea de puntos suspensivos hasta el final de la línea de tal forma que al final de ésta aparezca el segundo de los argumentos, es decir, la calificación obtenida, seguido de la calificación numérica en negrita y entre paréntesis.

En el caso de que tengamos las calificaciones en otro archivo y queramos importarlas desde dicho archivo, podremos cargar estas líneas con `\input{Archivo de datos}` o con `\include{Archivo de datos}`. Estos dos comandos pueden utilizarse, en general, para incluir cualquier archivo de texto en un documento LaTeX.

8.4. Creación de hipervínculos

Ya se ha comentado la posibilidad de crear páginas web con LaTeX. Tanto en ellas como en otros documentos "electrónicos", pueden ser útil introducir hipervínculos que nos dirijan de un lugar a otro del documento o de otros elementos.

Para obtener un archivo.PDF a partir del archivo.TEX, habitualmente utilizamos programas como PDFLaTeX y DVIPDFM. En cualquiera de los dos casos, se pueden incorporar tanto marcadores como enlaces de manera manual. Para ello, vamos a emplear comandos `\special` en el archivo.TEX. Algunos de los enlaces que se pueden crear con estos comandos son de suma utilidad, como pueden ser los que permiten enlazar las diferentes secciones desde el índice general o las referencias bibliográficas con las entradas en la lista de la bibliografía.

El paquete `hyperref` automatiza la generación de enlaces cuando se trata de referencias cruzadas usuales en LaTeX. La aplicación de los comandos `\special` dependerá de que la visualización vaya a realizarse con DVIPDFM o PDFLaTeX, por lo que debe indicársele al paquete cuál va a utilizarse.

Este paquete fue desarrollado en 1997 por S. Rathz y H. Oberediek con el fin, como ya hemos comentado, de crear de manera automática enlaces de hipertexto entre las referencias cruzadas de LaTeX.

Por "generación automática" se entiende que, tras cargar el paquete, los comandos de LaTeX que realizan las referencias cruzadas también crearán los enlaces. En consecuencia, no se van a requerir nuevas acciones para conseguir tanto la que hemos explicado en los capítulos anteriores como la de hipertexto.

Pero el paquete `hyperref` va a permitirnos también crear enlaces de hipertexto a documentos externos o a direcciones en Internet. Para ello, el paquete proporciona nuevos comandos específicos que permitirán llevar a cabo esas funciones.

A continuación indicamos algunos comandos que sirven para crear enlaces manualmente. Daremos la expresión del comando seguido de una breve descripción del mismo.

- \hyperlink{*Nombre*}{*Texto*} e \hypertarget{*Nombre*}{*Texto*}: El primer comando convierte el argumento *Texto* en un enlace para conectar con el destino denominado *Nombre* y que se ha definido con el segundo comando.
- \hyperimage{*ImagenURL*}: Incorpora la imagen que se indica en la dirección URL.
- \href{*URL*}{*Texto*}: Crea un enlace a la dirección *URL*.
- \hyperref{*URL*}{*Categoría*}{*Nombre*}: *Texto* se vuelve un enlace con la dirección *URL#Categoría.Nombre*.

Para más datos sobre la creación de hipervínculos y el funcionamiento del paquete `hyperref` puede verse [16].

8.5. BibTEX

BibTEX es un programa cuyo diseño corrió a cargo de Oren Patashnik. Su función es generar el entorno `thebibliography` con las entradas bibliográficas que se hayan citado a lo largo del documento de manera automática, buscando en bases de datos creadas especialmente para BibTEX.

BibTEX obtiene los datos de las referencias citadas en el documento que estamos componiendo de una base de datos y empleando un archivo de estilo para que LATEX sepa qué formato deben tener las lista de referncia.

Las bases de datos de BibTEX son archivos ASCII que emplean unas determinadas normas para indicar lo que va a ser un registro de la base de datos y cuáles los campos dentro de los registros. Estos archivos *base de datos* tienen extensión .BIB. Una página web que trata bastante bien este programa es [9].

8.5.1. Utilización de BibTEX

Para utilizar la base de datos `MiBase.bib` en el documento `MiArchivo.tex` solo hemos de seguir los siguientes cinco pasos:

1. Incluir antes de concluir el contenido del documento las órdenes

\bibliography{*NombreBase*}
\bibliographystyle{*Estilo*}

donde *NombreBase* es la base de datos a utilizar por BibTEX y *Estilo* es el estilo que BibTEX utilizará para generar la bibliografía. Los estilos habituales para BibTEX son los que indicamos a continuación:

plain: Las referencias se numeran siguiendo un orden alfabético por autor, luego por año y, en último lugar, por título.

unsrt: La diferencia con *plain* es que se numeran por orden de citación en el documento.

alpha: Las entradas se ordenan por autor, año y título, pero en vez de numerarlas se identifican las entradas con parte del nombre y el año de publicación.

abbrv: Es similar a *plain* salvo en que las entradas son más breves porque se abrevian los nombres de los autores, de los meses y de las revistas.

2. Incluir en nuestro documento el comando `\cite{`*Etiqueta*`}` allá donde quiera introducirse una referencia a algún documento incluido en el archivo `MiBase.bib`. *Etiqueta* debe ser la correspondiente al registro que queremos citar. Si se emplea `\nocite{`*Etiqueta*`}`, se imprime la referencia en la lista de bibliografía, pero sin imprimir una referencia a ella. El comando `\nocite*` al comienzo del documento produce como resultado la impresión del listado bibliográfico completo.

3. Compilamos el archivo `MiArchivo.tex` con el que estemos trabajando, con lo que el archivo `MiArchivo.aux` que se genera en la compilación guarda toda la información correspondiente a las citas bibliográficas que se realizan en el documento, además del nombre de la base de datos que se utilizará para generar la bibliografía y el estilo que se empleará para escribir la misma.

4. Para generar la lista bibliográfica, debemos previamente producir el entorno `thebibliography` mediante la ejecución del programa BibTEX sobre el archivo `MiArchivo.aux`. De este modo, generamos dos nuevos archivos llamados `MiArchivo.bbl` y `MiArchivo.blg`. El primero es el que contiene el entorno `thebibliography` producido por los comandos `\cite` y `\nocite` en el documento; mientras que el segundo contiene los errores que se han registrado durante la ejecución de BibTEX. El archivo `MiArchivo.bbl` tiene su información distribuida por bloques, en función de autor o de título, que se separan dentro de un mismo `\bibitem` por el comando `\newblock`.

5. El último paso que nos queda realizar es precisamente compilar nuevamente el archivo `MiArchivo.tex`, en el que se utilizará la información de los archivos con extensión .BBL y .AUX para escribir la lista bibliográfica y las citas oportunas en el texto.

8.5.2. Creación de bases de datos para BibTEX

Los archivos de BibTEX son archivos de texto que contienen las diversas referencias bibliográficas, que reciben el nombre de *registro*. Los registros se componen de diversos *campos*, que pueden ser necesarios, opcionales o ignorados (estos últimos no aparecen nunca en la referencia bibliográfica). La forma que posee uno de estos registros es la que aparece a continuación:

```
@ARTICLE{LamportIndex,
        AUTHOR = {Leslie Lamport},
        YEAR = 1987,
        TITLE = {Makeindex, {A}n Index Processor for {\LaTeX\ }},
        NOTE = {Documentaci{\'o}n contenida en makeindex.doc}}
```

Cuando introducimos un registro en una base de datos para BibTEX, debemos decidir en primer lugar cuál será el tipo de registro de esa referencia. El tipo de registro es la primera palabra detrás del signo @. En nuestro ejemplo, el tipo de registro es `ARTICLE`, ya que lo que se introduce es un artículo. Detrás del tipo se abre una llave que indica el comienzo de los datos que componen el registro. La primera palabra, `LamportIndex`, es la etiqueta que emplearemos en el comando `\cite` para citar esa obra.

Después de la etiqueta, aparecen los campos que componen el registro que se está editando. En el ejempo que nos ocupa se utilizan los campos `AUTHOR`, `YEAR`, `TITLE` o `NOTE` para definir el registro `ARTICLE`.

Campos de un registro

A continuación indicamos los campos que suelen ser de más utilizados:

`address:` Dirección del `publisher` o institución.

`author:` Los nombres de los autores se incluyen siguiendo reglas que indicamos en un próximo epígrafe.

`booktitle:` Título de un libro, parte del cual estamos citando.

`chapter:` Un número de capítulo, sección, etc.

`edition:` Edición de un libro. Se utilizan ordinales comenzados por mayúsculas.

`editor:` Nombre de los editores. Sigue las mismas reglas que el campo `AUTHOR`.

`institution:` Institución que financia un informe técnico (que corresponde al tipo `technical report`).

`journal:` Nombre de una revista.

`month`: Mes de publicación.

`note`: Permite introducir cualquier información adicional. La primera palabra debe comenzar con mayúsculas.

`number`: Número de una revista, informe técnico o trabajo de una serie.

`organization`: Organización que financia una conferencia o publica un manual.

`pages`: Una o más páginas o un intervalo de éstas. Los guiones sencillos (-) se convierten en guiones para denotar intervalos de páginas (–).

`publisher`: Nombre del que publica el documento referenciado.

`school`: Escuela, facultad o instituto donde se ha escrito una tesis.

`series`: Nombre de una serie o conjunto de libros.

`title`: Título del documento referenciado.

`type`: Tipo de informe técnico.

`volume`: Volumen de revista o de libro de varios volúmenes.

`year`: Año de publicación o de escritura (en trabajos no publicados).

Tipos de registros

Estos campos son necesarios u opcionales dependiendo del tipo de registro considerado. Seguidamente daremos una descripción de los diversos tipos de registro existentes:

`ARTICLE`: Artículo de una revista.

Necesario: `author, title, journal, year`.

Opcional: `volume, number, pages, month, note`.

`BOOK`: Libro de editorial especificada.

Necesario: `author` o `editor, title, publisher, year`.

Opcional: `volume` o `number, series, address, edition, month, note`.

`BOOKLET`: Trabajo impreso y distribuido sin nombre de editorial o institución que lo financie.

Necesario: `title`.

Opcional: `author, address, month, year, note`.

`CONFERENCE`: Igual que `IMPROCEEDINGS`.

`INBOOK`: Parte de un libro.

Necesario: `author` o `editor`, `title`, `chapter` o `pages`, `publisher`, `year`.

Opcional: `volume` o `number`, `series`, `type`, `address`, `edition`, `month`, `note`.

`INCOLLECTION`: Parte de un libro que tiene título propio.

Necesario: `author`, `title`, `booktitle`, `publisher`, `year`.

Opcional: `volume` o `number`, `series`, `type`, `address`, `edition`, `month`, `note`.

`IMPROCEEDINGS`: Artículo publicado en libro de actas.

Necesario: `author`, `title`, `booktitle`, `year`.

Opcional: `editor`, `volume` o `number`, `series`, `pages`, `address`, `publisher`, `organization`, `month`, `note`.

`MANUAL`: Documentación técnica.

Necesario: `title`.

Opcional: `author`, `organization`, `address`, `edition`, `month`, `year`, `note`.

`MASTERTHESIS`: Tesina (tesis de licenciatura).

Necesario: `author`, `title`, `school`, `year`.

Opcional: `type`, `address`, `month`, `note`.

`MISC`: Cualquier registro que no pueda catalogarse en los restantes.

Necesario:

Opcional: `author`, `title`, `month`, `year`, `note`.

`PHDTHESIS`: Tesis doctoral.

Necesario: `author`, `title`, `school`, `year`.

Opcional: `type`, `address`, `month`, `note`.

`PROCEEDINGS`: Actas de una conferencia.

Necesario: `title`, `year`.

Opcional: `editor`, `volume` o `number`, `series`, `address`, `month`, `note`, `publisher`, `organization`.

`TECHREPORT`: Informe publicado por escuela u otra institución.

Necesario: `author, title, institution, year`.

Opcional: `type, number, address, month, note`.

`UNPUBLISHED`: Documento con autor y título sin publicar formalmente.

Necesario: `author`, `title`, `note`.

Opcional: `month`, `year`.

Nombres en los campos `author` y `editor`

Los nombres que corresponden al campo `author` o `editor` deben ser escritos completos (que es lo más aconsejable) o como aparecen escritos en el artículo o libro a referenciar. En el caso de que haya más de un autor, éstos deben ir separados por la palabra `and`.

A la hora de introducir un autor con BibTEX son válidas las siguientes dos formas:

```
author = {Marta Robles}          author = {Robles, Marta}
```

siendo la mejor forma la segunda para los casos en los que hay dos apellidos. Esto se debe a que si le escribimos a BibTEX un autor como apellido utilizando la primera expresión:

```
author = {Manuel Ramos Mateos}
```

el programa BibTEX entiende `Ramos` como el segundo nombre y puede abreviar el nombre como `Manuel R. Mateos`. Para evitar estas situaciones, es aconsejable escribir los nombres empleando la otra variante:

```
author = {Ramos Mateos, Manuel}
```

Bibliografía

[1] **David Bausela**, *Los Primeros Pasos en LaTeXlandia*, actualizada el 8 de marzo de 2004. `http://usuarios.lycos.es/bausela/`

[2] **Bernardo Cascales Salinas, Pascual Lucas Saorín, José Manuel Mira Ros, Antonio Pallarés Ruiz y Salvador Sánchez-Pedreño Guillén.** *LaTeX. Una imprenta en sus manos.* Aula Documental de Investigación, Madrid, 2000.

[3] **CervanTeX**, actualizada el 29 de mayo de 2004. `http://www.cervantex.org/` o `http://filemon.mecanica.upm.es/CervanTeX/`

[4] **Patrick W. Daly**, *Graphics and Colour with LaTeX*, Max-Planck Institute für Aeronomie, actualizado el 4 de junio de 1998. `http://tex.loria.fr/graph-pack/grf/grf.htm`.

[5] **Ingo H. de Boer**, *WinShell for TeX*, actualizada el 6 de enero de 2004. `http://www.winshell.de/`

[6] **Michel Goossens, Frank Mittelbach and Alexander Samarin.** *The LaTeX Companion.* Addison-Wesley, 1994.

[7] **Frédéric Goualard**, *Manual for the* `prosper` *class*, actualizado el 28 de noviembre de 2000. `http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/prosper-doc.pdf`

[8] **Frédéric Goualard, Gregory Goualard, Janus Markus, Peter Møller Neergard and Didier Remy**, *Prosper: high quality slides in LaTeX*, actualizada el 14 de octubre de 2003. `http://prosper.sourceforge.net/`

[9] **Dana Jacobsen**, *The BibTeXFormat*, actualizado el 12 de diciembre de 1996. `http://www.ecst.csuchico.edu/~jacobsd/bib/formats/bibtex.html`.

[10] **Donald Ervin Knuth**, *Don Knuth's home page.* `http://www-cs-faculty.stanford.edu/~knuth/`

[11] **Leslie Lamport.** *LaTeX. A Document Preparation System. User's guide and manual references.* Addison-Wesley, Second Edition, 1994.

[12] **Leslie Lamport**, *Leslie Lamport's home page*, actualizada el 5 de mayo de 2004. `http://research.microsoft.com/users/lamport/`

[13] **Russel Lang**, *Ghostscript, Ghostview and GSview*, actualizada el 23 de febrero de 2004. `http://www.cs.wisc.edu/~ghost/`

[14] **MacKichan Software, Inc.**, *The LaTeX color package*, actualizada el 1 de agosto de 2003.
`http://www.mackichan.com/index.html?techtalk/475.htm~mainFrame`

[15] **MikTeX Project**, *MikTeX Project Page*, actualizada el 13 de julio de 2004. `http://wwww.miktex.org/`

[16] **Sebastian Rahtz and Heiko Oberdiek**, *Hypertext marks in LaTeX: a manual for* `hyperref`, actualizado en febrero de 2004. `http://www.tug.org/applications/hyperref/manual.html`

[17] **Aleksander Simonic**, *WinEdt Shell*, actualizada el 29 de mayo de 2004. `http://www.winedt.com/`