Introducción al LaTeX

Carlos Ivorra

LaTeX es un sistema de composición de textos que permite obtener fácilmente resultados de calidad profesional. Para asimilar correctamente la forma de trabajar con LaTeX conviene entender desde el primer momento que el LaTeX no es en absoluto un procesador de textos, sino un lenguaje de programación orientado a la generación de textos. Un procesador de textos es una aplicación informática que permite al usuario ir generando el texto deseado, mientras que las aplicaciones que nos permiten trabajar con LaTeX no son "el LaTeX", sino aplicaciones capaces de leer un "código fuente" escrito por el usuario en el lenguaje LaTeX e interpretarlo para generar un texto a partir de él.

Así, por ejemplo, si queremos que nuestro texto contenga la expresión $\sqrt{2}$, en nuestro código fuente deberemos escribir `\sqrt 2` y, cuando compilemos el código, veremos cómo se genera un documento en el que aparece la raíz cuadrada deseada. Vemos así que el código fuente y el texto generado tienen, en general, aspectos muy distintos.

Existen numerosas aplicaciones capaces de procesar código LaTeX. Este documento ha sido generado con TeXShop, que es una aplicación para el sistema operativo OS X de Apple, pero el lenguaje LaTeX es independiente de la plataforma en la que se ejecuta la aplicación que lo interpreta. Ocasionalmente haremos referencia a algunos aspectos específicos de TexShop, pero casi la totalidad del contenido de estas notas tiene que ver con el lenguaje LaTeX en sí y, por lo tanto, es independiente de la aplicación particular que se use o de la plataforma en la que se ejecute. Aquí supondremos que el lector tiene ya debidamente instalada en su ordenador y lista para ser usada una aplicación capaz de interpretar el lenguaje LaTeX.

Para entender el funcionamiento actual del LaTeX conviene conocer un poco de su historia. En 1978 Donald Knuth presentó la primera versión del lenguaje TeX, cuya naturaleza es la misma que acabamos de describir para el LaTeX, pero con la diferencia de que es un lenguaje de relativamente "bajo nivel", es decir, que requiere generar mucho código fuente para conseguir que el texto generado tenga alguna característica deseada. Sin embargo, entre las características que confieren al TeX su gran potencia se encuentra la de que permite generar "macros", es decir, "programas", bloques de código, que definen instrucciones complejas a partir de otras más simples, de modo que basta escribir una instrucción definida en una "macro" para obtener rápidamente un efecto que puede requerir muchas y complejas líneas de instrucciones TeX. En 1983 Leslie Lamport presentó la primera versión del LaTeX, que no es sino una "macro" gigantesca de TeX que define un lenguaje de "alto nivel" con el que resulta mucho más fácil el trabajo. La versión actual de LaTeX se llama LaTeX 2_ε y data de 1994.

1 Antes de empezar

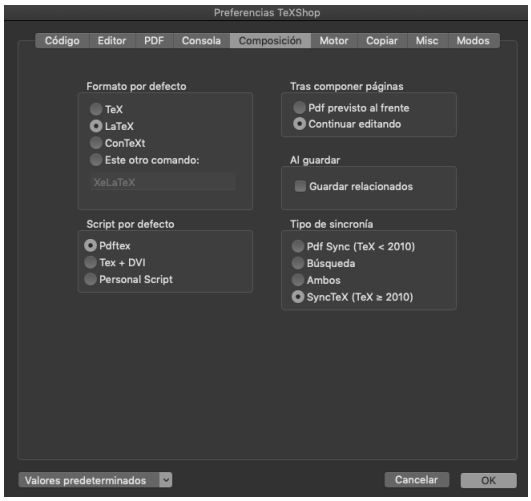
Antes de empezar a generar textos con LaTeX hay un par de decisiones que conviene tomar meditadamente y que pueden requerir una modificación de las opciones de la aplicación con la que trabajemos:

Pdftex vs. Tex+DVI El código fuente LaTeX puede escribirse con cualquier procesador de texto, y lo habitual es que su formato sea lo que se conoce como "texto sin formato" (plain text), aunque se guarda en archivos con la extensión tex. A partir de un archivo tex, los procesadores TeX y LaTeX originales generaban un archivo con extensión dvi, que contenía el texto generado en un formato específico. Con el tiempo, no tardaron en surgir aplicaciones auxiliares que transformaban los archivos dvi en archivos pdf, y éstas a su vez no tardaron en integrarse en las propias aplicaciones compiladoras de LaTeX.

Por eso, actualmente, la mayoría de las aplicaciones nos dan la opción de generar directamente un documento pdf a partir del código fuente o bien generar como antiguamente un documento dvi y, a partir de él generar un pdf. En general, la primera opción es preferible. La diferencia principal entre una y otra elección (aparte de que veamos aparecer o no un documento dvi junto a nuestro código fuente, el cual podemos borrar tranquilamente) radica en los tipos de figuras que LaTeXes capaz de incorporar a los documentos. El formato dvi requiere casi exclusivamente figuras en formato eps (poscript encapsulado), mientras que el uso de pdftex para generar directamente un pdf admite una gran cantidad de formatos de imagen (jpg, png, pdf etc.), pero requiere convertir los archivos eps en pdf.

Algunas aplicaciones generan imágenes de gran calidad en formato eps, pero esta calidad puede perderse al convertirlos a otros formatos. Si podemos obtener figuras de calidad en formato pdf u otros, será preferible usar la opción pdftex, mientras que si preferimos usar figuras eps (o si tenemos archivos antiguos con figuras en dicho formato), puede ser preferible usar la forma antigua de procesado.

Figura 1: Opciones sobre el motor de procesado



La figura 1 muestra cómo establecer en TeXShop el modo de procesado por defecto. En caso de que queramos contravenir esta opción para un documento en particular (aparte de podemos establecerlo manualmente mediante un menú) basta incluir al principio una de las "líneas mágicas" siguientes:

```
%!TEX TS-program = latex     %!TEX TS-program = Pdflatex
```

Codificación del texto Mucho más importante es la decisión que tomemos sobre el formato de texto de nuestro código fuente. Antiguamente, el formato estándar para los archivos de texto era el ASCII (American Standard Code for Information Interchange), consistente en que cada carácter de texto se representaba internamente en el ordenador como un número entre 0 y 127, lo que sólo permitía emplear unos pocos caracteres admisibles (los necesarios para escribir en inglés y poco más).

El TeX fue diseñado para que fuera posible generar textos arbitrariamente sofisticados a partir de un código fuente que no usara más que los 128 caracteres ASCII, y hoy en día sigue siendo posible escribir cualquier texto con esa limitación. Por ejemplo, si queremos escribir "engañó", podemos poner en el código fuente `enga\~n\'o`, de modo que todos los caracteres requeridos tienen un código ASCII asignado. Sin embargo, esto no es nada práctico.

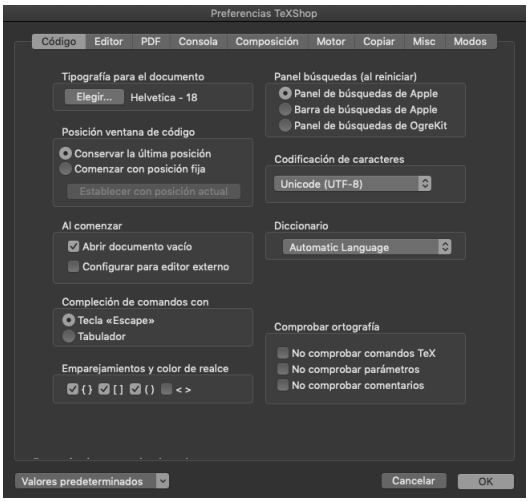
La limitación del código ASCII pronto resultó inadmisible para los sistemas operativos usuales, y el primer paso que se dio para superarla fue extender el código ASCII a 256 caracteres, que incluyeran signos usuales en muchos idiomas, como á, à, ñ, ç, etc. Por desgracia, esto no se hizo de forma unificada, sino que cada sistema operativo definió su propia extensión del código ASCII, de modo que había un código ASCII extendido de MSDOS, otro de Windows, otro de Macintosh, otro de Unix, etc.

El TeX permite convertir en instrucciones los caracteres extendidos que reconoce el sistema operativo, de modo que, por ejemplo, ó pueda convertirse en una instrucción TeX equivalente a `\'o`, y así el usuario puede escribir su código fuente usando los caracteres extendidos de forma natural. Sin embargo, el inconveniente es que el código fuente ya no es portable, sino que un archivo de texto con caracteres extendidos legible, por ejemplo, por una aplicación Windows, se leerá mal en una aplicación Macintosh, y viceversa.

Durante un tiempo esta falta de portabilidad resultó inevitable, pero actualmente ha surgido un nuevo estándar capaz de codificar, no ya los menos de 256 signos que caben en los códigos ASCII extendidos, sino millones de signos correspondientes a prácticamente todos los alfabetos conocidos. Se conoce como Unicode y, aunque sus orígenes se remontan a 1987, la primera versión estándar es de 1991. Existen distintas variantes, pero la más popular es UTF-8 (de Unicode Transformation Format), que tiene la ventaja de que los caracteres del ASCII estándar conservan su misma codificación, de modo que un texto ASCII estándar es también un texto UTF-8 estándar.

Los procesadores LaTeX modernos dejan elegir el formato en el que se guardará el código fuente y, a menos que el lector tenga un motivo poderoso para hacer otra elección (por ejemplo, por razones de compatibilidad con documentos antiguos), la opción más recomendable es sin duda UTF-8. La figura 2 muestra cómo establecer en TeXShop que los documentos se guarden en dicho formato.

Figura 2: Opciones sobre la codificación del texto



2 Cómo empezar

En este punto suponemos que el lector tiene ya debidamente configurada una aplicación capaz de compilar código LaTeX y que ha abierto un documento nuevo en blanco. ¿Ahora qué hay que hacer? Un código fuente mínimo que el lector puede probar a teclear es el siguiente:

```
\documentclass[a4paper,12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\begin{document}
¡Hola, mundo!


\end{document}
```

Si a continuación pulsa el botón de “componer" (o “typeset”) —que necesitará tener bien localizado en su aplicación—, debería obtener un texto que contenga únicamente la frase ¡Hola, mundo! En la carpeta que contiene el documento tex con el texto fuente se habrá generado un pdf con el mismo nombre y algunos archivos más de los que hablaremos más adelante. Analicemos el código:

El código fuente LaTeX debe estructurarse siempre en dos partes: un *preámbulo*, que empieza siempre con la instrucción `\documentclass`, y el *cuerpo* del documento, que empieza con `\begin{document}` y termina con `\end{document}`. En el preámbulo se incluyen las instrucciones que deben afectar globalmente a todo el documento, mientras que en el cuerpo se incluyen las instrucciones que generan el texto.

documentclass Según hemos dicho, la primera línea de un código LaTeX debe ser una instrucción `\documentclass[opciones]{clase}`.

La clase tiene que ser una de las muchas predefinidas en LaTeX y determinan la estructura global del documento. Las más básicas son `article` y `book`. Por ejemplo, un documento de clase *article* está preparado para ser dividido en secciones, mientras que un documento de clase `book` está preparado para ser dividido en capítulos, los cuales a su vez pueden tener varias secciones.

Las opciones van entre corchetes (y que sean opciones significa que podemos escribir simplemente `\documentclass{article}`). La primera establece el tamaño del papel, y la segunda el tamaño de la letra. Las únicas opciones válidas de tamaño son `10pt` (la opción por defecto, que no hace falta poner), `11pt` y `12pt`.

usepackage La instrucción `\usepackage[opciones]{paquete}` sirve para cargar paquetes adicionales que añaden nuevas funcionalidades al LaTeX básico.

El paquete `inputenc` define los caracteres no admitidos por el ASCII estándar para que LaTeX los entienda. Sin él habríamos obtenido un error a causa de la exclamación ¡, y también si hubiéramos tratado de escribir cualquier letra con un acento, o diéresis, etc. Como opción hay que indicarle la codificación del texto fuente. Las más habituales son:

`applemac`	ASCII extendido usado por Apple
`latin1`	ASCII extendido usado por Windows
`utf8`	UTF-8

Si el lector ha optado por una codificación de caracteres distinta de UTF-8, deberá usar la opción correspondiente.

Finalmente, el paquete `fontenc` tiene que ver con el tratamiento de las fuentes en el texto generado. El funcionamiento de las fuentes ha evolucionado mucho desde el origen del TeX. Sin entrar en detalles, omitir este paquete hará que las palabras acentuadas no se copien bien del pdf generado, o que LaTeX no sepa partir palabras a final de línea si llevan acentos, etc.

Aquí terminan los tecnicismos. Sin duda, el lector necesitará incorporar más paquetes a medida que vaya adentrándose en el uso del LaTeX, pero en general no debería copiar un preámbulo de otro documento e incluir en él muchas instrucciones cuya finalidad desconozca, sino que, por el contrario, es preferible no añadir nada sin un fin específico.

3 Escritura básica de texto

Espacios en blanco Los espacios en blanco (horizontales o verticales) en el código fuente no se corresponden con los que se obtienen en el texto generado. Hay que tener en cuenta lo siguiente:

- Uno o más espacios en blanco seguidos en el texto fuente producen un único espacio en el texto compilado. Si queremos obtener, por ejemplo, "triple espacio" debemos escribir `triple\␣\␣\␣espacio`, es decir, una barra \ seguida de un espacio en blanco genera un espacio acumulable a otros. Los espacios usuales no son acumulables.

- Con `~` se produce un espacio irrompible, que no se partirá a final de línea. Por ejemplo, conviene escribir `...desde~1 hasta~10` para evitar que un número quede a principio de línea.

- LaTeX deja un espacio extra después de un punto y seguido. Este efecto debe evitarse cuando el punto corresponde a una abreviatura. Por ejemplo, hay que escribir:

  ```
  ...en la página~1 y ss.\ se habla de...
  ```

  Sin embargo, cuando el punto está precedido de una mayúscula, LaTeX ya presupone que se trata de una abreviatura y deja tras él un espacio normal, con lo que no es necesario hacer esto. Por ejemplo:

  ```
  ...el Dr. D. S. Ramón y Cajal...
  ```

  Recíprocamente, si una frase termina en mayúscula hay que indicarlo para que LaTeX ponga el espacio adicional. Esto se consigue con `\@`. Por ejemplo:

  ```
  El partido más votado ha sido el P.P\@. En segundo lugar...
  ```

  Lo mismo sucede con el punto y coma, la interrogación, la exclamación o incluso con un punto seguido de paréntesis o comillas. Por ejemplo, hemos de escribir:

  ```
  Los cítricos (naranjas, etc.)\ tienen vitamina C\@.
  ```

- Un cambio de línea en el código fuente se traduce en un espacio en blanco (pero no un cambio de línea en el texto generado). Dos o más cambios de línea consecutivos dan lugar a un único cambio de línea en el texto generado.

- Podemos obtener espacios horizontales y verticales de cualquier longitud mediante las instrucciones

  `\hspace{4mm}` y `\vspace{5.6cm}`.

  La instrucción `\vspace` no funciona a principio de página para evitar que queden espacios en blanco indeseados debido a la paginación. Si queremos un espacio vertical a principio de página debemos escribir `\vspace*{5cm}`.

- Si queremos una separación estándar entre párrafos podemos emplear

  `\smallskip`, `\medskip` o `\bigskip`.

- Para una separación estándar entre palabras podemos emplear `\quad` o `\qquad`.

Signos ortográficos Con el uso del paquete `inputenc` podemos escribir directamente cualquier signo accesible desde el teclado, pero no está de más saber cómo pueden reducirse todos los caracteres al ASCII estándar (por ejemplo, esto puede ser necesario al escribir macros) y, por otra parte, hay algunos caracteres que no son accesibles desde el teclado. Los signos ortográficos más habituales se obtienen así:

```
á  \'a  à  \`a  â  \^a  ä  \"a
ñ  \~n  ¿  ?`   ¡  !`   ç  \c c
```

Hay muchos más. Por ejemplo, el matemático polaco Paul Erdős se escribe `Erd\H os`.

LaTeX tiene cuatro guiones distintos: `-` se usa entre palabras, `--` se usa entre números, `---` es el guión ortográfico y `\-` es el guión opcional que sólo aparecerá si la palabra tiene que ser partida a final de línea. Notemos la diferencia:

Físico-químico, páginas 4–8, Hay —de hecho— varios tipos...

Las comillas se consiguen con el acento grave y el apóstrofo:

'comillas simples'	`'comillas simples'`
"comillas dobles"	`''comillas dobles''`
«comillas francesas»	`<<comillas francesas>>`

Las comillas francesas sólo funcionan si en el preámbulo hemos puesto `\usepackage[T1]{fontenc}`

Podemos conseguir caracteres elevados como en "M[a] del Mar" escribiendo

```
M\textsuperscript{a} del Mar'
```

Lógicamente, también existe `\textsubscript`, pero no es tan útil.

Hay ciertos signos que son accesibles desde el teclado, pero que TeX se reserva para usos especiales. Para obtenerlos como meros signos hemos de usar los comandos siguientes:

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

Los puntos suspensivos se consiguen con `\ldots`.

Comentarios En cuanto escribimos un %, LaTeX deja de leer el resto de la línea. De este modo podemos insertar comentarios en el código fuente que no aparecerán en el texto generado, o suprimir temporalmente parte del código fuente.[1]

Tamaños, estilos y tipos de letra El aspecto del texto admite diversas modificaciones, y cada una de ellas se puede conseguir de varias formas más o menos equivalentes.

Tamaño Los tamaños disponibles son:

tiny, scriptsize, footnotesize, small, normalsize, large, Large,
LARGE, huge, Huge.

Se consiguen de este modo: `{\Large Texto}`. Notemos que las llaves abarcan incluso el comando, es decir, que no hay que escribir `\Large{Texto}`. Si lo hacemos así, el texto continuará en tamaño Large fuera de las llaves. En el ejemplo no se aprecia que `\huge` es mayor que `\LARGE` porque `huge` está en minúsculas. Cuando en `documentclass` se ha fijado la opción `12pt` no hay diferencia entre `\huge` y `\Huge`.

Cuando el texto alterado es largo, suele ser más útil generar un entorno (una porción de código delimitada por un `\begin{}` y un `\end{}`):

[1] No obstante, algunos editores, como TexShop, admiten "líneas mágicas", es decir, líneas que empiezan con % (de modo que LaTeX no las lee), pero que producen un efecto sobre el editor. Ya hemos señalado el caso de `%!TEX TS-program = Pdflatex`, que fuerza el uso de Pdflatex. Otro ejemplo es que `%:Etiqueta` genera una etiqueta que permite acceder rápidamente al punto del código donde está insertada desde un menú.

```
\begin{Large}
Texto grande
\end{Large}
```

Forma Las formas posibles son Recta, *Itálica*, *Inclinada* y MAYÚSCULAS PEQUEÑAS. La forma más simple de conseguirlas es

```
{\it Italic}, {\sl slanted} y {\sc Small Caps}.
```

Sin embargo, hay que advertir que estos comandos son comandos TEX, y que LATEX proporciona dos versiones alternativas más refinadas:

```
{\upshape Upright}  {\itshape Italic},
{\slshape slanted}  {\scshape Small Caps}.
```

O bien

```
\textup{Upright}  \textit{Italics}
\textsl{Slanted}  \textsc{Small Caps}
```

Notemos que `\itshape` requiere ser rodeada por las llaves, mientras que `\textit` precede a las llaves.

Serie La serie puede ser media o **negrita**. La instrucción TEXpara poner el texto en negrita es `{\bf Boldface}`, mientras que LATEX proporciona las alternativas

```
{\mdseries Medium}  {\bfseries Boldfase}
\textmd{Medium}     \textbf{Boldface}
```

Familia La familia puede ser Roman, `Sans Serif` y `Typewriter`. Pueden conseguirse mediante `{\rm Roman}`, `{\tt Sans Serif}` y `{\tt Typewriter}`, o bien con las alternativas LATEX:

```
{\rmfamily Roman}  {\sffamily Sans Serif}  {\ttfamily Typewriter}
\textrm{Roman}     \textsf{Sans Serif}     \texttt{Typewriter}
```

Como en el caso de los tamaños, también es posible crear entornos:

```
\begin{bfseries}
Texto en negrita
\end{bfseries}
```

con cualquiera de las opciones, y vale igualmente `bfseries` o `bf`.

La ventaja principal de las versiones LaTeX es que son acumulables. Por ejemplo, escribiendo `{\bfseries\itshape Texto}` obtenemos ***Texto***, con el texto a la vez en negrita y en cursiva. En cambio, `{\bf\it Texto}` produce *Texto*, donde sólo la última instrucción tiene efecto. No obstante, en los contextos habituales en los que no es necesario combinar estilos, las instrucciones TeX son más breves, luego más prácticas.

La instrucción `\em` alterna entre el texto recto y en cursiva. Por ejemplo:

```
{\em Estoy leyendo {\em Los miserables} por las noches.}
```

da lugar a *Estoy leyendo* Los miserables *por las noches.*

Finalmente, `\underline{texto subrayado}` genera <u>texto subrayado</u>.

Justificado Normalmente TEX justifica el texto por ambos lados, partiendo las palabras de la forma más adecuada. Si queremos el texto centrado, alineado por la izquierda o alineado por la derecha usamos respectivamente los entornos `center`, `flushleft` y `flushright`. Por ejemplo, para alinear por la derecha una porción de texto se escribe:

```
\begin{flushright}
Texto alineado por la derecha
\end{flushright}
```

En cualquier caso podemos forzar el final de una línea (sin justificarla) con una doble barra `\\`. Opcionalmente, podemos indicar el espacio vertical deseado hasta la línea siguiente: `\\[3cm]`.

Finales de línea La forma habitual de terminar una línea antes de llegar al margen derecho de la página es dejando una línea en blanco (o más) en el código fuente, lo que provoca un cambio de párrafo.

La instrucción `\par`
fuerza un cambio de párrafo aunque en el código fuente se continúe escribiendo en la misma línea. (Nótese la sangría antes de "fuerza".)

En cambio, la instrucción `\newline`
termina la línea sin cambiar de párrafo, como puede apreciarse en este mismo párrafo, donde no hay sangría antes de "termina".

Se consigue un efecto diferente con `\linebreak`, que termina la línea justificándola, aunque la justificación resulte forzada. Es útil cuando no nos gusta el punto exacto por el que LaTeX ha partido una línea y queremos que la parta por otro cercano, donde la justificación forzada no resulte antiestética. Al revés, `\nolinebreak` impide que el final de línea caiga en un punto en concreto.

Un término medio se consigue con `\linebreak[n]`, donde `n` es un número del 1 al 4. `\linebreak[4]` es un fin de línea forzoso equivalente a `\linebreak`, mientras que para $n = 1, 2, 3$ es una sugerencia de cambio de línea, más insistente cuanto mayor sea n. Entonces LaTeX hará una valoración entre el grado de insistencia y lo forzado que resulta el fin de línea y sólo lo llevará a cabo si lo considera razonable. Es útil si cabe la posibilidad de que en el futuro vayamos a cambiar los márgenes del texto y un cambio de línea que ahora nos parece razonable pudiera dejar de serlo con otros márgenes.

Texto enmarcado Con `\fbox{texto enmarcado}` obtenemos \fbox{texto enmarcado}. Para enmarcar un párrafo entero usamos `\parbox{longitud}{párrafo}`, así:

Esto es un párrafo restringido a una longitud de 7 cm, conseguido con el comando `\parbox{7cm}{Esto es...}`

\fbox{\parbox{7cm}{Este párrafo además está dentro de un \fbox{\parbox{7cm}{...}}}}

Partición de palabras LaTeX parte automáticamente las palabras cuando conviene al final de línea, pero los criterios de partición de palabras son distintos en cada idioma. Por ejemplo, en inglés se parte "dif-er-en-ti-a-ble", mientras que en castellano es "di-fe-ren-cia-ble". El paquete `babel` incorpora diversas opciones relacionadas con el idioma de escritura. Por ejemplo, si ponemos en el preámbulo

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

conseguiremos, entre otras cosas, que LaTeX parta las palabras según las normas del castellano. No obstante, si lo único que necesitamos es esto, es mucho más práctico incluir en el preámbulo el código:

```
\makeatletter
\language\l@spanish
\makeatother
```

Conviene entender su estructura: LaTeX usa la arroba como un signo reservado para comandos. Esto significa que podemos escribir libremente en el código fuente @ y obtendremos una arroba en el texto generado, pero no podemos escribir ningún comando como `\l@spanish` que incluya una arroba sin generar un error. Esto sirve para que el TeX y las macros TeX puedan tener sus variables y funciones internas sin que el usuario las pueda modificar accidentalmente. No obstante, si queremos ejecutar intencionadamente una instrucción que requiere arrobas, podemos hacerlo si la incluimos entre `\makeatletter` y `\makeatother` ("convierte a la arroba en una letra usual" y "conviértela en *otra cosa,* en un signo reservado", respectivamente).

Signos adicionales Hay muchos paquetes que proporcionan signos de texto adicionales. Destacamos el paquete `marvosym`, que se carga incluyendo en el preámbulo la instrucción

```
\usepackage{marvosym}
```

Con él podemos usar, por ejemplo, `\EUR` para obtener el signo € del euro. Notemos que hay que escribir "`el signo \EUR\ del euro`", con una barra tras la instrucción, para que quede un espacio en blanco. Esto vale igualmente para cualquier instrucción que proporcione texto, pues en ocasiones puede interesar que no quede ningún espacio en blanco a continuación.

El paquete proporciona una serie de signos muy variados, como ☺ (`\Smiley`) o ⛆ (`\NoIroning`).

4 Escritura básica de matemáticas

Existen dos modos matemáticos: *text* y `display`. El primero se usa para escribir fórmulas matemáticas insertadas en el texto y el segundo para escribir fórmulas centradas. Unas mismas instrucciones pueden dar resultados diferentes según el modo. Por ejemplo $\sum_{n=1}^{\infty} \frac{1}{2^n} = 1$ está en modo *text,* mientras que

$$\sum_{n=1}^{\infty} \frac{1}{2^n} = 1$$

está en modo *display.*

El primer ejemplo se obtiene con

```
$\sum_{n=1}^\infty\frac1{2^n}=1$
```

Los signos `$` marcan el principio y el final del modo matemático *text.* Si en vez de `$` escribimos `$$` obtenemos el segundo ejemplo. En realidad los dólares son la forma que tiene TeX de entrar y salir de los modos matemáticos y LaTeX tiene sus propias versiones, que son `\( \)` para el modo *text* y `\[ \]` para el modo *display,* pero lo habitual es usar dólares.

Todos los signos escritos en modo matemático están por defecto en cursiva. Para evitar que un mismo signo se imprima de formas distintas en contextos diferentes debemos escribir entre dólares cualquier signo matemático insertado en el texto, aunque sea una simple letra.

Por ejemplo, para obtener

> Diremos que un elemento p de un dominio íntegro D es *irreducible* si no es nulo ni unitario y no tiene más divisores que sus asociados y las unidades.

escribimos

```
Diremos que un elemento $p$ de un dominio íntegro $D$ es {\it irreducible}
si no es nulo ni unitario y no tiene más divisores que sus asociados y las
unidades.
```

Algunas instrucciones tienen un efecto distinto en modo matemático. Por ejemplo, `a'` da a' (apóstrofo o comilla) en modo no matemático y a' (prima) en modo matemático.

Podemos forzar a que una fórmula en modo *text* se imprima como si estuviera en modo *display* o viceversa mediante las instrucciones `\displaystyle` y `\textstyle`. Por ejemplo, el código

```
$$
\sum_{n=1}^\infty{\textstyle \frac1{2^n}}=1
$$
```

produce como resultado

$$\sum_{n=1}^{\infty} \tfrac{1}{2^n} = 1$$

donde la fracción está en modo *text* y el resto en modo *display.*

Espacios Los espacios en blanco carecen de valor en el modo matemático, de modo que añadir o quitar un espacio en el texto fuente nunca cambia el resultado. En cada caso el TEX determina la distribución más conveniente de los espacios. No obstante, hay ocasiones en las que conviene aumentar o reducir los espacios, para lo cual contamos con las instrucciones `\,` (que produce un espacio pequeño) y `\␣`(que produce un espacio normal). Además está `\!`, que produce un espacio negativo. Para añadir espacios mayores tenemos `\quad` y `\qquad`, que funcionan también en modo matemático. Por ejemplo, para escribir

$$dy = 2x\,dx$$

conviene poner `$$dy = 2x\,dx$$`, de modo que dx quede un poco más separado.

Subíndices y superíndices Para poner subíndices se usa la barra baja `_` y para superíndices el circunflejo `^`. Por ejemplo, `$a_{i,j}=3^i-b_{i_j}$` produce $a_{i,j} = 3^i - b_{i_j}$. No hay ningún problema en poner al mismo tiempo un subíndice y un superíndice: `$a_i^{j+1}$` da a_i^{j+1}, pero no se pueden poner dos subíndices o dos superíndices a la vez. Una prima cuenta como superíndice, Por ejemplo, `$a_i'$` da a_i', pero `a^i'` da error, porque TEX considera que hay dos superíndices. En cambio acepta `$a'^i$`, que da a'^i. No obstante, siempre podemos engañar al TEX escribiendo `$a^i{}'$`, que produce $a^{i\prime}$.

Fracciones y raíces Las fracciones se consiguen con el comando

```
\frac{numerador}{denominador}
```

Por ejemplo, si escribimos `$$1+\frac1{1+\frac15}$$` obtenemos

$$1+\frac{1}{1+\frac{1}{5}}$$

Para las raíces tenemos el comando `\sqrt[n]{radicando}`. En general, los argumentos de los comandos LATEX que van entre corchetes son opcionales. Por ejemplo, `$\sqrt[5]{-1}$` produce $\sqrt[5]{-1}$. La instrucción `$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$` produce

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

Subrayado y similares El comando `\underline` vale también en modo matemático, pero ahora tenemos también `\overline`, que produce una barra sobre el texto, como en la fórmula $\overline{A+\overline{B}}$, que sale de `\overline{A+\overline B}`. También están disponibles `\overrightarrow` y `\overleftarrow`, que producen, por ejemplo, $\overrightarrow{ac}$.

Dos comandos similares son `\underbrace` y `\overbrace`, que producen llaves bajo o sobre el texto. Un subíndice tras un `\underbrace` o un superíndice tras un `\overbrace` aparece como una etiqueta en la llave, como en $(\underbrace{a,\ldots,a}_{15})$, que se obtiene con el código `(\,\underbrace{a,\ldots, a}_{15}\,)`.

Signos matemáticos Tenemos a nuestra disposición un amplio inventario de signos matemáticos estándar. TeX los clasifica en diversos tipos según el espacio que debe dejar entre ellos.

Normales Son las letras y números, que pueden escribirse también fuera del modo matemático, como `$a$`.

Ordinarios Son como los normales, pero sólo existen en modo matemático. Entre ellos se encuentran las letras griegas:

α	`\alpha`	ζ	`\zeta`	λ	`\lambda`	ρ	`\rho`	ϕ	`\phi`
β	`\beta`	η	`\eta`	μ	`\mu`	ϱ	`\varrho`	φ	`\varphi`
γ	`\gamma`	θ	`\theta`	ν	`\nu`	σ	`\sigma`	χ	`\chi`
δ	`\delta`	ϑ	`\vartheta`	ξ	`\xi`	ς	`\varsigma`	ψ	`\psi`
ϵ	`\epsilon`	ι	`\iota`	π	`\pi`	τ	`\tau`	ω	`\omega`
ε	`\varepsilon`	κ	`\kappa`	ϖ	`\varpi`	υ	`\upsilon`		

Γ	`\Gamma`	Λ	`\Lambda`	Π	`\Pi`	Υ	`\Upsilon`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Σ	`\Sigma`	Φ	`\Phi`	Ω	`\Omega`
Θ	`\Theta`								

Destacamos también los siguientes:

$\|$	`|`	$\emptyset$	`\emptyset`	∇	`\nabla`	$\forall$	`\forall`	$\exists$	`\exists`
∞	`\infty`	∂	`\partial`	$\neg$	`\neg`	$\aleph$	`\aleph`		

Desde el teclado podemos entrar directamente |.

Operadores binarios Son signos que han de aparecer cercanos a los signos anterior y posterior. Destacamos los siguientes:

$\pm$	`\pm`	$\mp$	`\mp`	$\times$	`\times`	$\div$	`\div`	$\circ$	`\circ`	$\cdot$	`\cdot`
$\cap$	`\cap`	$\cup$	`\cup`	$\setminus$	`\setminus`	$\oplus$	`\oplus`	$\otimes$	`\otimes`	$*$	`\ast`
$\wedge$	`\land`	$\vee$	`\lor`						

Notemos que para escribir $2 \cdot 3 = 6$ no debemos poner `$2.3=6$`, pues resultaría $2.3 = 6$, sino `$2\cdot 3 = 6$`.

Relaciones Son signos que han de quedar algo separados de los que les rodean. Los más importantes son $+, -, /, <, >, =$, que pueden introducirse directamente desde el teclado, además de los comandos siguientes:

$\leq$	`\leq`	$\geq$	`\geq`	$\equiv$	`\equiv`	$\sim$	`\sim`	$\simeq$	`\simeq`
$\mid$	`\mid`	$\parallel$	`\parallel`	$\subset$	`\subset`	$\subseteq$	`\subseteq`	$\supset$	`\supset`
$\supseteq$	`\supseteq`	$\approx$	`\approx`	$\in$	`\in`	$\ni$	`\ni`	$\notin$	`\notin`
$\neq$	`\neq`								

Observemos que `\neq` y `\notin` producen las negaciones de $=$ y `\in`. Para las demás relaciones podemos conseguir su negación anteponiendo `\not`. Por ejemplo, `$a \not\equiv b$` produce $a \not\equiv b$.

No hay que confundir los signos ordinarios `|` y `\|` con las relaciones `\mid` y `\parallel`. Producen los mismos signos, pero el espaciado es distinto. Por ejemplo, para obtener $|a + b| \leq |a| + |b|$ hemos de escribir `$|a+b|\leq |a|+|b|$` y no

```
$\mid a+b\mid \leq \mid a\mid + \mid b\mid$
```

que produciría $\mid a+b \mid \leq \mid a \mid + \mid b \mid$.

Entre las relaciones se encuentran también las flechas. Las más importantes son:

$\leftarrow$	`\leftarrow`	$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\Uparrow`
$\rightarrow$	`\rightarrow`	$\longrightarrow$	`\longrightarrow`	$\downarrow$	`\downarrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Downarrow$	`\Downarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`	$\updownarrow$	`\updownarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`	$\nearrow$	`\nearrow`
$\searrow$	`\searrow`	$\swarrow$	`\swarrow`	$\nwarrow$	`\nwarrow`

Operadores Son los signos que deben unirse al signo que sigue. Entre ellos se encuentran las funciones matemáticas, como

`\arccos`	`\cos`	`\sec`	`\ln`	`\lim`	`\max`	`\sup`
`\arcsin`	`\sin`	`\csc`	`\log`	`\ker`	`\min`	`\inf`
`\arctan`	`\tan`	`\cot`	`\exp`	`\det`	`\dim`	`\arg`

También se incluyen aquí los llamados "operadores grandes", que cambian de tamaño según el modo text / display. Entre ellos están:

$\sum$	$\displaystyle\sum$	`\sum`	$\cup$	$\displaystyle\bigcup$	`\bigcup`	$\oplus$	$\displaystyle\bigoplus$	`\bigoplus`	$\int$	$\displaystyle\int$	`\int`
$\prod$	$\displaystyle\prod$	`\bigcap`	$\cap$	$\displaystyle\bigcap$	`\prod`	$\otimes$	$\displaystyle\bigotimes$	`\bigotimes`	$\oint$	$\displaystyle\oint$	`\oint`

Para especificar los límites de una integral definida se usan los comandos de subíndice y superíndice. Por ejemplo, si escribimos `$$\int_a^b\!\!f(x)\,dx$$` obtenemos

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

Las sumas y las integrales son casos típicos donde conviene usar espacios negativos. Lo mismo vale con los límites. Para obtener:

$$\lim_{x\to x_0} f(x)$$

basta escribir `$$\lim_{x\rightarrow x_0}f(x)$$`. Sin embargo, en modo *text* el resultado es $\lim_{x\to x_0} f(x)$. En general, TEX trata de evitar que en una línea de texto sobresalgan cosas. Si, pese a todo, queremos forzar que la flecha quede debajo podemos escribir `$\lim\limits_{x\rightarrow x_0}f(x)$`. En general, la instrucción `\limits` delente de un subíndice o un superíndice y después de un operador hace que éste aparezca debajo o arriba del signo anterior en lugar de a la derecha.

Por ejemplo, podemos conseguir $\overset{c}{\underset{b}{a}}$ mediante `\mathop{a}\limits_b^c`. El primer comando convierte en operador a la a. Si no lo ponemos obtenemos un error.

Recíprocamente, `\nolimits` hace que los subíndices y los superíndices se comporten del modo habitual en el modo *display.*

Con `\smash{}` evitamos que una fórmula insertada en el texto, como $\int_a^b x^2\,dx$ separe las líneas anterior o siguiente como en este párrafo.

El código fuente es `$\smash{\displaystyle \int_a^bx^2\,dx}$`. Sin `\smash` el resultado habría sido éste: $\int_a^b x^2\,dx$. Aquí no podemos poner `\smash` sin que la integral invada las líneas anterior y posterior, pero a veces —como antes— es posible hacerlo.

Puntuación Además de los puntos suspensivos `\ldots`, en modo matemático disponemos también de `\ddots`, que produce unos puntos suspensivos, no a la altura de la línea, sino centrados. Compárese

$$a_1 + \ldots + a_n \qquad \text{con} \qquad a_1 + \cdots + a_n.$$

También tenemos puntos suspensivos verticales $\vdots$ y diagonales `\ddots`, que se obtienen con `\vdots` y `\ddots`, respectivamente, y que son útiles al construir matrices.

Delimitadores Son los signos que actúan a modo de paréntesis. Todos tienen una versión izquierda y una versión derecha. Los más importantes son (en su versión izquierda): $($, $[$, $\{$, $|$, $\|$, $\langle$. El paréntesis, el corchete y la barra vertical se introducen directamente desde el teclado, pero debemos recordar que las llaves requieren una barra: `\{`, pues TeX las usa para agrupar cosas. Los dos últimos se obtienen con `\|` y `\langle` (y `\rangle` para la versión derecha).

Las instrucciones `\left` y `\right` precediendo a unos delimitadores hacen que su tamaño se ajuste al del texto que encierran. Por ejemplo, si escribimos

```
$$\left(\frac{e^x+e^{-x}}2\right)^2$$
```

obtenemos

$$\left(\frac{e^x+e^{-x}}{2}\right)^2$$

También hemos de usar `\left` y `\right` cuando queramos usar un delimitador izquierdo a la derecha, o viceversa. Por ejemplo, para escribir $a \in]0,+\infty[$ conviene poner `a\in \left]0,+\infty\right[` para que el espaciado sea correcto. En otro caso saldría $a \in]0,+\infty[$, donde el $\in$ se mete dentro del corchete inicial.

Un `\left` no compensado con un `\right` produce un error, pero podemos usar `\left.` o `\right.` para poner un delimitador "invisible", si es que no queremos compensar otro.

Acentos Los modos matemáticos admiten más acentos que en modo no matemático, y los acentos comunes se obtienen con comandos distintos:

$\hat{a}$	`\hat a`	$\tilde{a}$	`\tilde a`	$\bar{a}$	`\bar a`	$\dot{a}$	`\dot a`	$\grave{a}$	`\grave a`
$\check{a}$	`\check a`	$\vec{a}$	`\vec a`	$\breve{a}$	`\breve a`	$\ddot{a}$	`\ddot a`	$\acute{a}$	`\acute a`

Los comandos `\widehat` y `\widetilde` producen versiones "anchas" de estos dos acentos, como en $\widehat{a+b}$.

Texto entre matemáticas Para introducir pequeñas porciones de texto entre expresiones matemáticas podemos usar `\mbox{texto}` o `\textrm{texto}`. Por ejemplo

$$\{x \in A \mid x > y \text{ para todo } y \in B\}$$

se obtiene con `$$\{x\in A\mid x>y \mbox{ para todo } y\in B\}$$`. Notemos que hay que dejar espacios en blanco dentro de las llaves.

Tamaños, estilos y tipos de letra En modo matemático las letras aparecen por defecto en cursiva, pero podemos elegir cualquiera de los estilos del modo no matemático con las mismas instrucciones `\rm`, `\it`, `\bf`, `\ss`, `\tt`, aunque si queremos combinarlas tendremos que usar las versiones LaTeX, que en modo matemático son `\mathrm`, `\mathit`, etc.

Todas estas instrucciones afectan sólo a letras, números y letras griegas mayúsculas. Por ejemplo, si escribimos `$\mathbf{a+\alpha=4}$` obtenemos $\mathbf{a}+\alpha=\mathbf{4}$, donde sólo la a y el 4 aparecen en negrita. Si queremos una fórmula entera en negrita hemos de usar `{\boldmath $fórmula$}`. Por ejemplo, `{\boldmath $a+\alpha=4$}` produce $\boldsymbol{a+\alpha=4}$. Es importante que `\boldmath` ha de usarse fuera del modo matemático, y su efecto es que todo el texto en modo matemático que aparezca en los límites de la declaración aparezca en negrita. Si sólo queremos un signo en negrita dentro de una fórmula hemos de usar `\mbox`. Por ejemplo, `$x+\mbox{\boldmath{$\nabla$}f}$` produce $x+\boldsymbol{\nabla}$f.

Junto a los estilos matemáticos `\displaystyle` y `\textstyle` existen otros dos, `\scriptstyle` y `\scriptscriptstyle`, en los que LaTeXentra automáticamente cuando escribe subíndices y subsubíndices, respectivamente, aunque también se usan en otros contextos, como en fracciones dentro de fracciones. Así, si escribimos

```
$$\sqrt 5 = 1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{\ddots}}}}$$
```

obtenemos

$$\sqrt{5} = 1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{\ddots}}}}$$

donde el LaTeX ha ido reduciendo el estilo paulatinamente, pero el resultado no es satisfactorio. En este caso queda mejor si forzamos a que todos los términos de la fracción continua mantengan el estilo display, mediante

```
$$\sqrt 5 = 1+\frac{1}{\displaystyle1+\frac{1}{\displaystyle1
+\frac{1}{\displaystyle1+\frac{1}{\ddots}}}}$$
```

que produce

$$\sqrt{5} = 1+\frac{1}{\displaystyle1+\frac{1}{\displaystyle1+\frac{1}{\displaystyle1+\frac{1}{\ddots}}}}$$

Ya hemos visto las instrucciones `\left` y `\right`, que ajustan el tamaño adecuado de un delimitador. A veces LaTeX no sabe calcular el tamaño adecuado y entonces hemos de hacerlo directamente mediante las instrucciones TeX `\bigl`, `\Bigl`, `\biggl`, `\Biggl`, que producen delimitadores izquierdos de distintos tamaños, y las corespondientes `\bigr`, etc., que producen los correspondientes delimitadores derechos. Así podemos obtener, por ejemplo, $\Big||a|+|b|\Big|$. Con la opción `12pt` algunas de estas instrucciones producen el mismo efecto.

El modo matemático tiene un estilo adicional llamado *caligráfico,* que vale sólo para letras mayúsculas. Se obtiene con `\mathcal{Mayúsculas}`. Por ejemplo, `$\mathcal PX$` produce $\mathcal PX$. También puede usarse `\cal`, pero `$\cal PX$` produce $\cal PX$. Para conseguir lo mismo con `\mathcal` necesitamos poner `\mathcal{PX}`.

Fuentes adicionales Hay varios paquetes que proporcionan fuentes adicionales para los modos matemáticos. Aquí destacaremos unos pocos de ellos. Por ejemplo, poniendo en el preámbulo

```
\usepackage{eucal}
```

las letras caligráficas cambian a una fuente más elegante. Por ejemplo, así `$\mathcal PX$` produce $\mathcal PX$ en lugar de $\mathcal PX$. Si no queremos reemplazar unas fuentes por otras, podemos usar

```
\usepackage[mathscr]{eucal}
```

y entonces las nuevas fuentes caligráficas se llaman con `\mathscr` en lugar de `\mathcal`.

Los paquetes `calrsfs` y `mathrsfs` proporcionan un juego alternativo de letras caligráficas, que se llaman, respectivamente, con `\mathcal` y `\mathscr`.

Con `\usepackage{amsfonts}` obtenemos las fuentes góticas `\mathfrak`, como en $\mathfrak{Goethe}$ y las fuentes de "pizarra" (sólo mayúsculas), como $\mathbb R$, $\mathbb N$, etc., que se obtienen con `\mathbb R`, etc.

Con `\usepackage{bbm}` y usando `\mathbbm R` se obtiene $\mathbbm R$, $\mathbbm N$, etc. También proporciona las variantes `\mathbbss` y `\mathbbtt`, que dan lugar a $\mathbbss R$ y $\mathbbtt R$, respectivamente.

Signos adicionales Hay muchos paquetes que proporcionan signos adicionales. Destacamos el paquete `amssymb`. Se carga poniendo en el preámbulo:

```
\usespackage{amssymb}
```

y automáticamente disponemos de una larga lista de signos adicionales. He aquí algunos ejemplos:

$\varnothing$ `\varnothing` $\subsetneq$ `\subsetneq` $\nmid$ `\nmid` $\trianglelefteq$ `\trianglelefteq`

Numeración y referencia a ecuaciones Para que una ecuación aparezca numerada, en lugar de escribirla entre dólares dobles usaremos el entorno

```
\begin{equation}\label{suma}
2+2=5
\end{equation}
```

El resultado es

$$2 + 2 = 5 \tag{1}$$

de modo que al escribir `La ecuación (\ref{suma}) no es exacta.` obtenemos:

La ecuación (1) no es exacta.

Vemos así que no conviene hacer referencia directamente al número de una ecuación, sino que es mejor ponerle una etiqueta mediante `\label{etiqueta}` y luego citar la ecuación mediante `(\ref{etiqueta})`. De este modo, LaTeX pondrá el número correcto aunque éste varíe debido a que hayamos insertado o borrado ecuaciones previas.

Para que aparezca el número correcto en una ecuación es necesario compilar dos veces el documento. La primera vez aparece (??) en su lugar. La razón es que la primera vez que compilamos LaTeX toma nota de las etiquetas en un archivo auxiliar que tiene el mismo nombre que nuestro documento .tex, pero con la extensión .aux. La segunda vez que compilamos lee en el archivo .aux la numeración que corresponde a cada etiqueta. Cuando se produce algún cambio de numeración, o un error de compilación, o se borra el archivo .aux, el número puede desaparecer, por lo que al dar un documento por terminado hay que asegurarse de que se muestran las referencias a las ecuaciones, y compilar una segunda vez si no es el caso.

5 Formato de página

Márgenes Para modificar los márgenes del texto conviene cargar el paquete `anysize`, poniendo en el preámbulo

```
\usepackage{anysize}
```

Esto permite poner, también en el preámbulo, la instrucción

```
\marginsize{izquierdo}{derecho}{superior}{inferior}
```

que establece las medidas de los cuatro márgenes. Éstas pueden especificarse en cualquiera de las unidades que reconoce LaTeX. Por ejemplo `3cm`, `35mm`.

Las dimensiones del papel se especifican como opción en la instrucción `\documentclass`. La opción más habitual es `a4paper`. Hay otras opciones de interés:

11pt, 12pt Tamaño de la letra (la opción por defecto es `10pt`).

twoside Hace intercambia los márgenes derecho e izquierdo en las páginas pares e impares, de modo que el texto impreso a doble cara se superponga.

landscape Hace que el texto se imprima en el papel apaisado.

twocolumn Hace que el texto se divida en dos columnas por página.

leqno, fleqn La primera numera las ecuaciones por la izquierda, la segunda alinea las ecuaciones por la izquierda en modo *display*.

La opción `landscape` hace que el texto se imprima en el papel apaisado. Por ejemplo, un documento LaTeX puede empezar así:

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

Cabeceras y pies de página También podemos poner en el preámbulo la instrucción `\pagestyle{opción}`, con una de las opciones siguientes:

empty Las cabeceras y pies de página quedan en blanco.

plain Cabecera en blanco y pie de página con el número de página en el centro.

headings La cabecera y el pie contienen información dependiente del tipo de documento especificado en la instrucción `documentclass`.

myheadings Las cabeceras contienen el texto especificado mediante las instrucciones:

`\markboth{cabecera izquierda}{cabecera derecha}`

o

`\markright{cabecera derecha}`

Si no hemos especificado la opción `twoside` en `documenclass` todas las páginas se consideran derechas. Las instrucciones `\markboth` y `\markright` no van necesariamente en el preámbulo, sino que pueden ir en el cuerpo del documento, y así permiten cambiar las cabeceras en cualquier momento.

Se puede cambiar el estilo de una página en particular con `\thispagestyle{opción}`.

Es posible incluir notas al pie como ésta[2] mediante `\footnote{Texto al pie}`.

Paginación LaTeX distribuye el texto en páginas como considera oportuno, pero podemos forzar que una página termine y el texto continúe en la página siguiente mediante el comando `\newpage`.

El comando `\pagebreak` termina también la página, pero mantiene el justificado vertical aunque resulte forzado. Como en el caso de `\linebreak`, existe la versión `\pagebreak[n]`, donde `n` es un número del 1 al 4 que permite a LaTeX sopesar si termina o no la página en función del grado de insistencia y de lo forzada que resultará la justificación vertical.

Para documentos en dos columnas estas instrucciones sólo provocan cambios de columna y no de página.

Podemos modificar la numeración de las páginas mediante `\setcounter{page}{27}`, que hace que la página actual pase a ser la número 27.

Otros ajustes Las magnitudes siguientes determinan el aspecto del documento:

oddsidemargin Distancia del texto al borde izquierdo del papel menos una pulgada. Si el estilo distingue entre páginas pares e impares, esta longitud sólo afecta a las impares. Las páginas pares las regula `\evensidemargin` (pero si el estilo no distingue las páginas pares y las impares entonces `\evensidemargin` no tiene efecto).

textwidth Anchura del texto.

[2] Esto es una nota al pie.

topmargin Distancia de la cabecera al borde superior del papel menos una pulgada.

headheight Altura de la cabecera.

textheight Altura del texto.

footskip Distancia del texto al pie de página.

Ejemplo:
`\setlength\textheight{10cm}` Ajusta la altura del texto a 10 cm.
`\addtolength\textheight{-5cm}` Resta 5cm a la altura del texto.

Estas magnitudes son globales, es decir, que afectan a todo el documento. En cambio, las magnitudes siguientes son locales, en el sentido de que pueden modificarse en el cuerpo del documento para que afecten sólo a algunos fragmentos del texto:

`\hoffset`	Margen izquierdo menos una pulgada.
`\size`	Ancho de línea.
`\leftskip`	Espacio adicional a principio de línea (0 por defecto).
`\rightskip`	Espacio adicional a final de línea (0 por defecto).
`\parindent`	Longitud del sangrado.
`\parfillskip`	Espacio de relleno en la última línea de un párrafo.
`\baselineskip`	Distancia entre las líneas base de dos líneas consecutivas.
`\parskip`	Espacio vertical entre dos párrafos.
`\abovedisplayskip`	Espacio por encima de una fórmula centrada.
`\belowdisplayskip`	Espacio por debajo de una fórmula centrada.

La sintaxis para modificar estas magnitudes es `\baselineskip=.5cm`

`\noindent` suprime el sangrado de la línea siguiente.

`\hangafter=n` añade una sangría adicional (independiente de la que de `\parindent`) a partir de la línea `n` del párrafo y cuya longitud se especifica mediante `hangindent = 5mm`. Si `n` es negativo la sangría se aplica a las primeras líneas del párrafo. Si `\hangindent` es negativo la sangría se aplica a la derecha.

`\parshape = n` $\mathtt{i}_1\mathtt{l}_1 \cdots \mathtt{i}_n\mathtt{l}_n$ produce un párrafo en el que las `n` primeras líneas tienen sangría $\mathtt{i}_1$ y longitud $\mathtt{l}_k$.

El primer párrafo tras un título de sección no se sangra por defecto, para que sangrarlo usamos el paquete `\usepackage{indentfirst}`.

6 El formato artículo

Discutimos aquí algunas de las características que dependen de haber seleccionado específicamente el formato `article` en la instrucción `\documentclass`, aunque todas ellas valen con mínimas variantes para las otras clases básicas, como `book` o `report`.

El título En el preámbulo del documento podemos especificar la información siguiente:

```
\title{título}
\auctor{autor}
\date{fecha}
```

Para que aparezcan impresos debemos usar la instrucción `\maketitle` después del `\begin{document}`. Si no queremos que aparezca ninguna fecha habremos de especificar `\date{}`, o de lo contrario aparecerá la fecha de hoy. Si el título es largo y queremos cortarlo por algún punto en concreto podemos usar `\\`.

Si hay varios autores hemos de separarlos con `\and`. Si uno o varios autores deben llevar una nota al pie de agradecimientos, etc., ésta se incluye mediante la instrucción `\thanks{Agradecimientos}`. Por ejemplo, una declaración de autores puede ser:

```
\author{J. López\thanks{Financiado en parte por...}\and
J. García\thanks{El segundo autor desea agradecer...}}
```

El abstract El abstract se escribe en un entorno delimitado por `\begin{abstract}` y `\end{abstract}`. El resultado es:

> **Abstract**
>
> Este documento pretende ser una introducción al LaTeX, en el que discutimos sus características más básicas.

La palabra "Abstract" la pone LaTeX automáticamente. Si queremos que ponga otra cosa, por ejemplo, "Resumen", sólo tenemos que incluir en el preámbulo la instrucción

```
\renewcommand{\abstractname}{Resumen}
```

Secciones Un documento en formato artículo puede organizarse mediante las instrucciones siguientes:

`\part` `\section` `\subsection` `\subsubsection` `\paragraph` `\subparagraph`

Por ejemplo, la cabecera de esta sección se ha obtenido con la instrucción:

```
\section{El formato artículo}\label{formatoart}
```

El número 6 lo pone LaTeX automáticamente. La primera frase de este párrafo se ha obtenido mediante:

```
El número \ref{formatoart} lo pone \LaTeX\ automáticamente.
```

Vemos así que podemos hacer referencia a las secciones de un documento con el mismo sistema de etiquetas válido para las ecuaciones. Mediante `\pagerref{etiqueta}` obtenemos el número de página donde se encuentra la etiqueta. Por ejemplo, si escribimos:

```
Véase la sección \ref{formatoart} en la página \pageref{formatoart}.
```

obtenemos: Véase la sección 6 en la página 20.

Podemos alterar la numeración automática de las secciones. Por ejemplo, si queremos que después de la subsección 3.4 venga la subsección 3.7 (por ejemplo, porque la subsección 3.6 va a escribirla un amigo nuestro), antes de iniciar la subsección 3.7 escribimos `\setcounter{subsection}6`, y así LaTeX "se creerá" que ya está en la subsección 3.6, y al iniciar una nueva le asignará el número 7.

La instrucción `\appendix` hace que las secciones siguientes se consideren apéndices y, en lugar de ser distinguidas con números, se distinguen con letras.

Índice La instrucción `\tableofcontents` genera una sección con un índice de contenidos. Mediante `\setcounter{tocdepth}4` especificamos la profundidad del índice. Con un 0 se reflejan únicamente las partes, con un 1 las secciones, etc.

Bibliografía La bibliografía al final del artículo se especifica de la forma siguiente:

```
\begin{thebibliography}{X}
\bibitem{Cer} M. de Cervantes, el Ingenioso Hidalgo...
\bibitem{Gon} L. de Góngora, Soledades.
\bibitem{Que} Fco. de Quevedo, Historia del Buscón llamado...
\end{thebibliography}
```

El resultado es:

References

[1] M. de Cervantes, el Ingenioso Hidalgo...

[2] L. de Góngora, Soledades.

[3] Fco. de Quevedo, Historia del Buscón llamado...

El argumento `X` es cualquier porción de texto de longitud mayor o igual que el mayor número que vaya a aparecer en la lista de referencias. Por ejemplo, si va a haber entre 10 y 99 referencias hay que poner al menos dos caracteres, para que LaTeX sepa cuánto espacio debe dejar por la izquierda para que los números queden alineados por la derecha.

Si queremos cambiar "References" por otra cosa incluimos en el preámbulo

```
\renewcommand{\refname}{Bibliografía}
```

Para referirnos a una obra, por ejemplo a la de Cervantes, usaremos la instrucción `\cite{Cer}`, o incluso `\cite[pp.\ 20--21]{Cer}`, lo cual produce [1, pp. 20–21].

Si no queremos que las referencias aparezcan numeradas, sino con etiquetas, sólo hemos de indicarlas entre corchetes al lado de cada `\bibitem`. Por ejemplo:

```
\begin{thebibliography}{X}
\bibitem{Ce} M. de Cervantes, el Ingenioso Hidalgo...
\bibitem{Go} L. de Góngora, Soledades.
\bibitem{Qu} Fco. de Quevedo, Historia del Buscón llamado...
\end{thebibliography}
```

produce:

References

[C1605] M. de Cervantes, el Ingenioso Hidalgo...

[G1613] L. de Góngora, Soledades.

[Q1626] Fco. de Quevedo, Historia del Buscón llamado...

Y la cita `\cite{Go}` produce [G1613].

Teoremas Es posible generar entornos para enunciar teoremas, corolarios, etc. de modo que LaTeX los numere automáticamente y nos permita hacer referencia a ellos mediante etiquetas. El formato es el siguiente:

```
\newtheorem{teo}{Teorema}[section]
```

Esta orden (que conviene situar en el preámbulo) define un entorno llamado `teo`, de modo que cuando escribimos

```
\begin{teo}
Hay infinitos números primos.
\end{teo}
```

obtenemos

Teorema 6.1 *Hay infinitos números primos.*

Así pues, "teo" es el nombre del entorno que hemos de poner tras el `\begin` y el `\end`, "Teorema" es la palabra que aparece en el texto compilado y el argumento opcional "section" hace que el número de cada teorema aparezca precedido por el número de la sección actual (y vuelve a 1 al cambiar de sección). Si queremos definir un entorno "Corolario" de modo que la numeración de los corolarios sea correlativa a la de los teoremas escribiremos en el preámbulo

```
\newtheorem{cor}[teo]{Corolario}
```

y así, al escribir

```
\begin{cor}
El conjunto de los números primos no está acotado.
\end{cor}
```

obtenemos

Corolario 6.2 *El conjunto de los números primos no está acotado.*

Con `\newtheorem{cor}{Corolario}` habríamos obtenido **Corolario 1**, mientras que con `\newtheorem{cor}{Corolario}[section]` habríamos obtenido **Corolario 6.1**, con una numeración independiente de la de los teoremas.

El texto de un teorema aparece siempre en itálica. Si no lo queremos así usamos `\rm`. Si queremos poner nombre a un teorema lo hacemos entre corchetes. Por ejemplo,

```
\begin{teo}[Euclides]
Hay infinitos números primos.
\end{teo}
```

produce

Teorema 6.3 (Euclides) *Hay infinitos números primos.*

Una etiqueta `\label{lo que sea}` tras un `\begin{teo}` permite referirse al teorema mediante `\ref{lo que sea}`. Podemos alterar la numeración de los teoremas con `\setcounter`. Por ejemplo, si hacemos `\setcounter{teo}{12}` y volvemos a escribir el teorema anterior obtenemos

Teorema 6.13 (Euclides) *Hay infinitos números primos.*

Otros formatos de documento La diferencia principal entre el formato `article` y el formato `book` es que éste admite una sección `\chapter` intermedia entre `\part` y `\section`, así como que no admite abstract. Por su parte, el formato `report` admite `\chapter` y `\abstract`.

Los formatos se definen en archivos con extensión `cls`, de modo que si ponemos en la misma carpeta que nuestro documento un archivo `miestilo.cls` podremos poner en el preámbulo `\documentclass{miestilo}`. Así, muchas editoriales proporcionan documentos `cls` para que los documentos LaTeX que los usan tengan automáticamente el formato deseado.

7 Matrices, tablas, listas, etc.

Veamos ahora algunas instrucciones que permiten organizar el texto de formas un poco más sofisticadas.

Matrices Consideremos la matriz:

$$A = \left(\begin{array}{rccl} 1.234 & -5 & x & 0.234 \\ 280 & 0 & x^2+2 & 1.22 \end{array} \right)$$

Ha sido obtenida con el texto fuente siguiente:

```
$$
A=\left(
\begin{array}{rccl}
1.234&-5&x&0.234\\
280&0&x^2+2&1.22
\end{array}
\right)
$$
```

Al poner `\begin{array}{rccl}` indicamos que vamos a escribir una matriz con cuatro columnas, de las cuales la primera está justificada por la derecha, las otras dos por el centro y la última por la izquierda. Dentro de cada fila, las distintas entradas se separan mediante signos `&` y el final de cada fila se indica mediante `\\`. Notemos que `array` no pone los paréntesis, sino que éstos los ponemos nosotros antes y después de la matriz.

Las estructuras matriciales pueden usarse para conseguir fórmulas que no representan realmente matrices. Por ejemplo, si escribimos

```
$$
f(x) = \left\{
\begin{array}{cl}
x^2+y&\mbox{si } x > y,\\
y^3&\mbox{si } x\leq y.
\end{array}\right.
$$
```

obtenemos

$$f(x) = \left\{ \begin{array}{cl} x^2+y & \text{si } x > y, \\ y^3 & \text{si } x \leq y. \end{array} \right.$$

Notemos el uso de `\right.` para compensar el delimitador izquierdo `\left\{` sin que se escriba nada realmente. Otro ejemplo de escritura matricial es

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (x,y) & \mapsto & x^2+y \end{array}$$

obtenida mediante

```
$$
\begin{array}{rccl}
f:&\mathbb R^2&\longrightarrow&\mathbb R\\
&(x, y)&\mapsto&x^2+y
\end{array}
$$
```

Si entre las "erres","ces" y "eles" que determinan las columnas escribimos `@{algo}`, el texto que figure en 'algo' se escribirá entre las columnas correspondientes y, si hay comandos, éstos se ejecutarán en cada fila. Por ejemplo, si consideramos que las columnas del ejemplo anterior están demasiado separadas podemos insertar espacios negativos entre ellas para obtener

$$\begin{array}{rccl} f: & \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ & (x,y) & \mapsto & x^2+y \end{array}$$

El texto fuente es

```
$$
\begin{array}{r@{\hspace{-2pt}}c@{\hspace{-4pt}}
c@{\hspace{4pt}}l}
f:&\mathbb R^2&\longrightarrow&\mathbb R\\
&(x, y)\ \ &\mapsto&x^2+y
\end{array}
$$
```

Hemos dejado un par de espacios tras (x, y) para desplazarlo hacia la derecha respecto al R2 que tiene encima. Puede parecer extraño que un espacio positivo de cuatro puntos acerque las dos últimas columnas. La razón es que LaTeX añade un cierto espacio entre las columnas de una matriz, pero al poner una @ este espacio se suprime. Por lo tanto `@{\hspace{4pt}}` resta el espacio extra y suma 4 puntos, y el resultado es negativo. Un simple `@{}` ya reduce el espacio entre columnas.

Una forma de alterar el espaciado de varias columnas a un tiempo es mediante la instrucción `@{\extracolsep{3mm}}`. El efecto es un espacio extra de 3mm entre TODAS las columnas que siguen (salvo que pongamos otra instrucción de este tipo más adelante). Posteriores @ no anulan este espacio extra. Recordemos que la distancia entre filas se puede alterar con `\\[2mm]`. A menudo queremos poner líneas verticales entre las columnas de una matriz. En principio deberíamos poner `@{|}`, pero dada la frecuencia de uso puede abreviarse en `|`. Por ejemplo, si escribimos

```
$$
\left(
\begin{array}{c|ccc}
a&0&\cdots &0\\
\hline
0&1 & & \\
\vdots & &\ddots & \\
0& & & 1
\end{array}
\right)
$$
```

obtenemos

$$\left(\begin{array}{c|ccc} a & 0 & \cdots & 0 \\ \hline 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \end{array}\right)$$

Notemos que `\hline` produce una línea horizontal. Debe ponerse antes de la primera fila o después de \\. Si ponemos `||` entre dos columnas obtendremos una doble barra vertical. También podemos poner varios `\hline` seguidos.

Tablas El entorno `tabular` es idéntico a array salvo que se usa en modo normal (no matemático). Todo lo dicho anteriormente para `array` vale aquí y todo lo nuevo que diremos aquí vale también para `array`.

Podemos unir varias columnas en una mediante `\multicolumn`. Consideremos por ejemplo la tabla siguiente:

	Precio
Producto	Mín–Máx
A	100–300
B	1 230–2 000
C	3 000–5 000

Observamos que la palabra "Precio" se extiende sobre la segunda y la tercera columna. Para conseguirlo, tras dejar en blanco la primera posición de la primera fila escribimos `\multicolumn{2}{c|}{Precio}`, cuyo efecto es que la palabra "precio" ocupe las dos columnas siguientes en posición centrada y con una línea vertical a la derecha.

Entre la primera y la segunda fila hay una línea horizontal que se extiende sólo entre la segunda y la tercera columna. Esto se consigue con `\cline{2-3}`. El texto fuente completo es:

```
\begin{tabular}{|c|r@{--}l|}
\hline
&\multicolumn{2}{c|}{Precio}\\
\cline{2-3}
Producto &Mín &Máx \\
\hline
A &100 & 300 \\
B &1\,230 & 2\,000 \\
C &3\,000 & 5\,000 \\
\hline
\end{tabular}
```

Podemos usar `\multicolumn{1}{r}{algo}` para escribir "algo" en una entrada pero con un justificado distinto del correspondiente a la columna. A veces queremos introducir texto en una entrada de una tabla pero no queremos que la anchura de la columna sea la del texto, sino que éste se parta en líneas. Para ello, al declarar la columna correspondiente no ponemos `r`, `l` o `c`, sino `p{4cm}`, donde la longitud indica la anchura de la columna.

Por ejemplo, la tabla:

Producto	Precio Mín–Máx	Observaciones
A	100–300	Es el más barato de todos.
B	1 230–2 000	Presenta la mejor relación calidad-precio.
C	3 000–5 000	Producto de super-extra-mega-lujo.

se consigue con

```
\begin{tabular}{|c|r@{--}l||p{4cm}|}
\hline
&\multicolumn{2}{c||}{Precio}& \\
\cline{2-3}
Producto &Mín&Máx&\multicolumn{1}{c|}{Observaciones}\\
\hline
A&100 & 300 & Es el más barato de todos.\\
B&1\,230 & 2\,000& Presenta la mejor relación calidad-precio.\\
C&3\,000 & 5\,000& Producto de super-extra-mega-lujo.\\
\hline
\end{tabular}
```

A veces queremos que una tabla tenga una anchura prefijada, por ejemplo la de la página. Esto se consigue con el entorno `\tabular*`, que es idéntico a `tabular` salvo por que admite como argumento opcional la anchura de la tabla. Si ponemos `\textwidth` obtenemos una tabla cuyo ancho es el de la página. En tal caso hemos de indicar entre qué columnas queremos que se inserte el espacio extra para cuadrar la tabla. Esto se consigue con `@{\extracolsep{\fill}}`. Por ejemplo, la tabla

Año	A	B	C	D	E	Mín–Máx
1996	1 000	250	400	2 130	300	250–2 130
1997	1 230	200	800	2 000	500	200–2 000
1998	1 600	220	700	2 100	1 500	220–2 100

tiene el ancho de la página y éste se consigue rellenando el espacio entre las columnas 2–3, 3–4, 4–5 y 5–6. El texto fuente es

```
\noindent\begin{tabular*}{\textwidth}
{r|c@{\extracolsep{\fill}}cccc|@{\extracolsep{0mm}\ }r@{--}l|}
Año&A&B&C&D&E&Mín&Máx\\
\hline
1996&1\,000&250&400&2\,130&300&250&2\,130\\
1997&1\,230&200&800&2\,000&500&200&2\,000\\
1998&1\,600&220&700&2\,100&1\,500&220&2\,100\\
\hline
\end{tabular*}
```

Alineación de ecuaciones Para alinear ecuaciones podemos usar estructuras matriciales. No obstante, en el caso más simple conviene usar los entornos `eqnarray` y `eqnarray*`. Ambos son equivalentes a un entorno `\begin{array}{rcl}`, con la peculiaridad de que las columnas primera y tercera aparecen en estilo display mientras que la segunda en estilo texto. No hay que poner dólares para usarlos. Por ejemplo, si escribimos

```
\begin{eqnarray*}
x&=&y\\
x^2&=&xy\\
x^2-y^2&=&xy-y^2\\
(x+y)(x-y)&=&y(x-y)\\
x+y&=&y\\
2y&=&y\quad \mbox{(por la primera ecuación)}\\
2&=&1
\end{eqnarray*}
```

obtenemos

$$\begin{aligned}
x &= y \\
x^2 &= xy \\
x^2-y^2 &= xy-y^2 \\
(x+y)(x-y) &= y(x-y) \\
x+y &= y \\
2y &= y \quad \text{(por la primera ecuación)} \\
2 &= 1
\end{aligned}$$

Si suprimimos los asteriscos obtenemos las ecuaciones numeradas:

$$\begin{aligned}
x &= y && (1)\\
x^2 &= xy && (2)\\
x^2-y^2 &= xy-y^2 && (3)\\
(x+y)(x-y) &= y(x-y) && (4)\\
x+y &= y && (5)\\
2y &= y \quad \text{(por la primera ecuación)} && (6)\\
2 &= 1 && (7)
\end{aligned}$$

Si no queremos numerar alguna ecuación usamos `\nonumber`. Por ejemplo, mediante

```
\begin{eqnarray}
x&=&y\label{prime}\\
x^2&=&xy\nonumber\\
x^2-y^2&=&xy-y^2\nonumber\\
(x+y)(x-y)&=&y(x-y)\nonumber\\
x+y&=&y\nonumber\\
2y&=&y\quad \mbox{por (\ref{prime})}\nonumber\\
2&=&1\nonumber
\end{eqnarray}
```

obtenemos

$$\begin{aligned} x &= y \\ x^2 &= xy \\ x^2-y^2 &= xy-y^2 \\ (x+y)(x-y) &= y(x-y) \\ x+y &= y \\ 2y &= y \quad \text{por (1)} \\ 2 &= 1 \end{aligned} \tag{1}$$

Otras estructuras matriciales A menudo es útil `\shortstack`, que produce una tabla (en modo no matemático) de una sola columna. Por ejemplo, podemos obtener

$$\lim_{\substack{(x,y)\to(0,0)\\ y=mx}} f(x,y)$$

mediante

```
$$
\lim_{\shortstack{$\scriptstyle (x, y)\rightarrow (0, 0)$\\
$\scriptstyle y = mx$}}f(x, y)
$$
```

Notemos que con `\shortstack` volvemos a modo no matemático, por lo que hemos de poner dólares en cada columna y, más aún, hemos de pasar a tamaño de subíndice. Por defecto el texto aparece centrado, pero podemos escribir `\shortstack[l]{texto}` si lo queremos alineado por la izquierda o con una `r` si lo queremos por la derecha.

El macro básico para trabajar con TEX es el llamado Plain TEX. El LATEXincorpora algunas de sus instrucciones, entre las cuales figuran varias sobre estructuras matriciales. La sintaxis es completamente distinta. Por ejemplo, otra forma de conseguir matrices es con `\matrix`. Así, podemos obtener una matriz con columnas centradas como

$$\matrix{3&52&300\cr 41&2&x^2}$$

mediante `$$\matrix{3&52&300\cr 41&2&x^2}$$`. Si en lugar de `\matrix` usamos `\pmatrix` obtenemos los paréntesis:

$$\pmatrix{3&52&300\cr 41&2&x^2}$$

Una opción más interesante es `\bordermatrix`, que produce una fila y una columna

fuera de los paréntesis, como en

$$\begin{array}{c} \\ j \\ \\ k \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & j & & k & & \\ \end{array} \\ \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \end{array}$$

que ha sido obtenida con

```
$$
\bordermatrix{& & &j& &k\cr
&1\cr
& &\ddots\cr
j& & &0& &1\cr
& & & &\ddots\cr
k& & &1& &0\cr
& & & & & &\ddots\cr
& & & & & & &1}
$$
```

También es de ayuda el comando `\cases`, que produce una matriz de dos columnas, la primera en modo matemático y la segunda en modo no matemático, con una llave abierta a la izquierda, como en

$$f(x) = \begin{cases} x^2 & \text{si } x > 0, \\ 3 & \text{si } x \leq 0, \end{cases}$$

obtenida mediante `$$f(x) = \cases{x^2&si $x > 0$,\cr 3&si $x\leq 0$,}$$`

El entorno tabbing Hay otro entorno que produce texto alineado en columnas de forma similar a los tabuladores de una máquina de escribir. Se entra y sale de él mediante `\begin{tabbing} ... \end{tabbing}` en modo no matemático. Se cambia de línea mediante `\\`. Cada vez que se sitúa un `\=` se fija un stop y el comando `\>` salta al stop siguiente. Por ejemplo,

Nombre Apellido Teléfono
Juan Gómez 3141592
Pedro Saenz 2718281

se consigue con

```
\begin{tabbing}
Nombre \= Apellido \= Teléfono\\
Juan \> Gómez\> 3141592\\
Pedro \> Saenz\> 2718281
\end{tabbing}
```

Una línea acabada con \kill en lugar de \\ no se imprime, pero los stops que contiene se conservan. Por ejemplo, si escribimos

```
\begin{tabbing}
Nombrexxxxx\= Apellidoxxxxxxxxxx \= Teléfono\kill
Nombre\> Apellido \> Teléfono\\
Juan \> Gómez\> 3141592\\
Pedro \> Saenz\> 2718281
\end{tabbing}
```

obtenemos

Nombre	Apellido	Teléfono
Juan	Gómez	3141592
Pedro	Saenz	2718281

El comando \+ hace que las líneas siguientes empiecen en el stop siguiente al previsto. Por ejemplo, mediante

```
\begin{tabbing}
margen xxxxx\=Nombrexxxxx\= Apellidoxxxxxxxxxx
\= Teléfono\+\kill
Nombre\> Apellido \> Teléfono\\
Juan \> Gómez\> 3141592\\
Pedro \> Saenz\> 2718281
\end{tabbing}
```

obtenemos

Nombre	Apellido	Teléfono
Juan	Gómez	3141592
Pedro	Saenz	2718281

Los comandos \+ se pueden acumular y a su vez pueden ser contrarrestados con \-. El comando \< salta al stop anterior. Al principio de una línea contrarresta localmente el efecto de un \+ (pero en las líneas siguientes sigue actuando el \+).

El comando \' hace que el texto precedente se justifique por la derecha respecto al stop anterior. Por ejemplo,

```
\begin{tabbing}
xxxxxxxxxxxxx\=\kill
20\> 30\' 500\\
100\> 300\' 8000
\end{tabbing}
```

produce

20	30	500
100	300	8000

El comando `\‘` lleva el texto que sigue contra el margen derecho. Por ejemplo,

```
\begin{tabbing}
Nombrexxxxx\=\kill
Nombre \> Apellido \‘Teléfono\\
Juan \> Gómez \‘3141592\\
Pedro \> Saenz \‘2718281
\end{tabbing}
```

produce

Nombre	Apellido	Teléfono
Juan	Gómez	3141592
Pedro	Saenz	2718281

Listas Para enumerar o clasificar párrafos disponemos de los tres entornos `enumerate`, `description` e `itemize` (para entrar en ellos escribimos `\begin{enumerate}` etc.) Cada párrafo se inicia con el comando `\item`. Por ejemplo, si escribimos:

```
Orden del día:
\begin{enumerate}
\item Lectura y aprobación del acta anterior.
\item Asuntos de tercer ciclo.
\item Ruegos y preguntas.
\end{enumerate}
```

el resultado es:

Orden del día:

1. Lectura y aprobación del acta anterior.
2. Asuntos de tercer ciclo.
3. Ruegos y preguntas.

Podemos referirnos indirectamente a los puntos de la forma habitual, por ejemplo, si ponemos

```
\item\label{TC} Asuntos de tercer ciclo.
```

después podremos escribir:

```
La documentación referente al punto \ref{TC} está disponible en Secretaría.
```

con el efecto:

La documentación referente al punto 7 está disponible en Secretaría.

Si cambiamos `enumerate` por `itemize` el resultado es:

Orden del día:

- Lectura y aprobación del acta anterior.
- Asuntos de tercer ciclo.
- Ruegos y preguntas.

Con `description` no aparece ninguna marca al comienzo de cada párrafo. En cualquiera de los tres entornos, `\item` puede llevar como argumento opcional la etiqueta que queramos que aparezca. Por ejemplo,

G. Rossini Il barbiere di Siviglia, La gazza ladra, La cenerentola, Semiramide, Il viaggio a Reims, Guillaume Tell.

G. Puccini Tosca, La Boheme, Madama Butterfly, Turandot, Manon Lescaut, La fanciulla dal west.

G. Verdi Nabucco, Un ballo in maschera, Rigoletto, Il trovattore, La traviatta, La forza del destino, Otello, Aida, Falstaff.

se obtiene con:

```
\item[G. Rossini] Il barbiere di Siviglia,...
\item[G. Puccini] Tosca,...
\item[G. Verdi] Nabucco,...
```

El resultado con `itemize` es similar:

G. Rossini Il barbiere di Siviglia, La gazza ladra, La cenerentola, Semiramide, Il viaggio a Reims, Guillaume Tell.

G. Puccini Tosca, La Boheme, Madama Butterfly, Turandot, Manon Lescaut, La fanciulla dal west.

G. Verdi Nabucco, Un ballo in maschera, Rigoletto, Il trovattore, La traviatta, La forza del destino, Otello, Aida, Falstaff.

Podemos modificar el estilo de enumerate. Por ejemplo, si ponemos en el preámbulo:

```
\makeatletter\renewcommand\theenumi{\@alph\c@enumi}\makeatother
\renewcommand\labelenumi{\theenumi)}
```

Las etiquetas serán a), b), c) ... en lugar de 1., 2., 3. ... Con más detalle: donde pone `alph` podemos poner:

arabic Produce números 1, 2, 3, ... (no es necesario especificarlo, es la opción por defecto).

alph Produce letras a, b, c, ...

Alph Produce letras mayúsculas A, B, C, ...

Roman Produce números romanos I, II, III, IV, ...

roman Produce romanos en minúsculas i, ii, iii, iv, ...

El argumento de `\labelenumi` contiene los signos adicionales que acompañan al número (representado por `\theenumi`). Por ejemplo, si queremos los números entre corchetes [1], [2], [3], ... especificaremos

```
\renewcommand\labelenumi{[\theenumi]}
```

8 Objetos flotantes y figuras

Objetos flotantes son porciones de texto que no pueden cortarse al terminar la página. LaTeX reconoce dos tipos: tablas y figuras. Si queremos que una tabla quede al principio o al final de la página no podríamos hacerlo tecleándola sin más, pues no sabemos en qué punto de la página aparece cada cosa que escribimos en el texto fuente. Para conseguirlo usamos el entorno `table`. Por ejemplo, la tabla **??** que aparece al final de la página ha sido generada mediante:

```
\begin{table}[b]
\caption{Las 16 categorías de signos \TeX}\label{etiqueta}
\begin{center}
\begin{tabular}{|c|lc||c|lc|}
Categ.&Significado& Ejemplo&Categ.&Significado& Ejemplo\\
.....................................................
\end{tabular}
\end{center}
\end{table}
```

El argumento opcional `[b]` (bottom) indica que la tabla debe ir al final de la página. Otras alternativas son `t` (top, arriba), `h` (here, aquí) y `p` (page, en una página aparte, toda con tablas y figuras). Dentro de un entorno table podemos poner una o más tablas (que aparecerán todas en bloque). Cada una puede llevar su `\caption{título}`, lo que produce además la numeración automática de la tabla y la posibilidad de referirse a ella mediante etiquetas.

En principio LaTeX no escribe "Tabla", sino "Table". Para traducirlo ponemos en el preámbulo: `\def\tablename{Tabla}`.

Es importante comprender que lo único que hace el entorno `table` es buscar espacio para la tabla, pero no crea ninguna tabla. La tabla se crea normalmente con un entorno `tabular` o `tabbing` o de cualquier otro modo. También es cosa nuestra distribuir las tablas (si es que hay varias) una al lado de otra, una bajo la otra o como sea, así como regular la distancia entre el título y la tabla etc.

Tabla 1: Las 16 categorías de signos TeX

Categ.	Significado	Ejemplo	Categ.	Significado	Ejemplo
0	Carácter de escape	\	8	Subíndice	_
1	Inicio grupo	{	9	Ignorado	
2	Fin grupo	}	10	Espacio	␣
3	Modo matemático	$	11	Letra	a, b, c, ...
4	Tabulador	&	12	Otro	@
5	Fin de línea		13	Activo	
6	Parámetro	#	14	Comentario	%
7	Superíndice	^	15	Inválido	

Si estamos escribiendo a dos columnas (con la opción `twocolumn`) las tablas ocupan la página entera. Si queremos una tabla que sólo ocupe una columna usamos el entorno `table*`.

El entorno `figure` se comporta exactamente igual que `\table`, sólo que en los títulos pone "figura" en lugar de "tabla". Mejor dicho, pone "figure". Si queremos traducirlo usamos `\def\figurename{Figura}`.

Consideremos, por ejemplo, la figura siguiente:

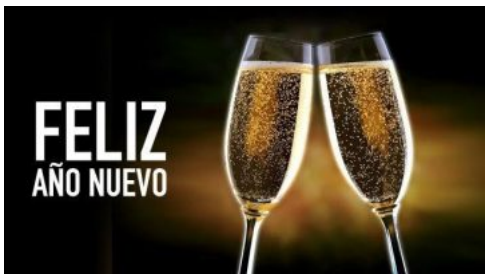


Figura 1: Felicitación

Ha sido insertada con el código siguiente:

```
\begin{figure}[h]
\begin{center}
\includegraphics[scale=.5]{copas.jpg}
\caption{Felicitación}
\end{center}
\end{figure}
```

La instrucción `\includegraphics` es la forma usual de insertar figuras en el documento. Para usarla hay que cargar el preámbulo el paquete `graphicx`, mediante `\usepackage{graphicx}`. La instrucción `\includegraphics` admite varios argumentos opcionales entre corchetes. Los más importantes son los siguientes:

`angle=45`	Gira la imagen, en este caso 45 grados en sentido antihorario.
`height=3cm`	Fija la altura de la imagen.
`scale=0.9`	Escala la imagen, a un 90% de su tamaño en el ejemplo.
`width=40mm`	Fija la anchura de la imagen.

Para combinar varias imágenes o superponer texto es útil el entorno `picture`. Por ejemplo, consideremos la figura siguiente:

Ha sido generada con el código siguiente:

```
\setlength\unitlength{1mm}
\begin{center}
\begin{picture}(80,40)
\put(4,4){\includegraphics[scale=.5]{copas.jpg}}
\put(70,7){Copa}
\put(70,22){Burbujas}
\put(70,10){\vector(-1,1){11}}
\put(69,23){\vector(-1,0){16}}
\put(0,8){\rotatebox{90}{Texto vertical}}
\put(20,0){Texto horizontal}
\end{picture}
\end{center}
```

El entorno `picture` requiere que se indique entre paréntesis la anchura y la altura que se desea reservar para la imagen, pero hay que escribir sólo números (como el `(80,40)` del ejemplo) sin unidades. La unidad puede especificarse en cualquier momento previo (por ejemplo en el preámbulo) mediante la instrucción `\setlength\unitlength{1mm}`.

Dentro de un entorno `picture` podemos usar `\put(x,y){objeto}` para situar un objeto en la posición de coordenadas `(x, y)`, entendiendo que las coordenadas `(0,0)` corresponden al extremo inferior izquierdo de la figura (aunque es posible poner coordenadas negativas).

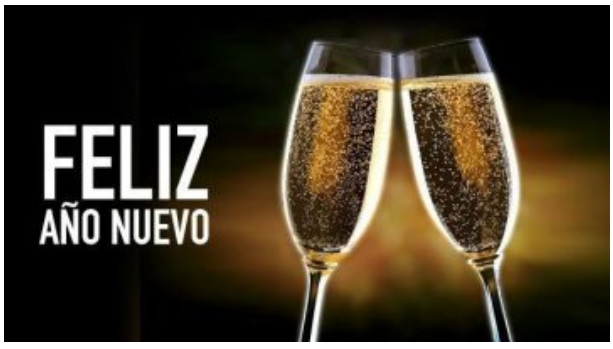
El comando `\rotatebox{ángulo}{texto}` está definido en el paquete `graphicx` y permite girar un texto los grados especificados. Es independiente del entorno `picture`, es decir, que puede usarse igualmente fuera de él.

Dentro de un entorno `picture` podemos usar también los comandos `\vector(a,b)(l)` y `\line(a,b)(l)`, que generan flechas y segmentos, respectivamente. Los enteros `a`, `b` tienen que estar entre -4 y 4 y ser primos entre sí. El segmento o flecha resultante tendrá pendiente b/a y su extremo final será el inicial más `(la,lb)`.

También es posible dibujar circunferencias y círculos mediante `\circle{diámetro}` y `\circle*{diámetro}`.

Hay también instrucciones para dibujar óvalos y otras curvas, pero para gráficos más complicados suele ser más práctico usar algún programa de dibujo y luego insertar la figura mediante `\includegraphics`.

Una forma de dejar espacio en el texto para figuras, tablas, etc. es mediante el entorno `minipage`. Así:




```
\noindent\begin{minipage}{7cm}
Una forma de...
\end{minipage}\qquad
\begin{minipage}{8cm}
\includegraphics[width=8cm]{...}
\end{minipage}
```

La longitud dada es la anchura de la minipágina.

Podemos decidir si varias minipáginas consecutivas se alinean por arriba, por abajo o por el centro. Para ello incluimos un parámetro opcional así: `\begin{minipage}[t]{8cm}`, que puede ser `[t]` (top) `[b]` (bottom). Si no se pone nada, la minipágina queda centrada en la línea.

También es posible especificar una altura de la minipágina, y en tal caso también podemos indicar si el contenido debe situarse en la parte superior, en la inferior o centrado. La sintaxis es:

```
\begin{minipage}[t][altura][b]{anchura}
```

La `[t]` indica que la minipágina se alineará por su parte superior, mientras que la `[b]` indica que el contenido de la minipágina se situará en su parte inferior.

9 Definiciones

LaTeX es en realidad un lenguaje de programación, por lo que tiene muchas capacidades que van más allá de la mera escritura de texto. No vamos a entrar en ello, pero sí que conviene saber que es posible introducir definiciones, ya sea para simplificar la escritura de expresiones más complejas, ya sea para modificar algunas opciones.

Por ejemplo, si no queremos teclear `\mathbb R` cada vez que queremos escribir $\mathbb{R}$, sólo tenemos que escribir en cualquier parte del documento (preferiblemente en el preámbulo) `\def\R{\mathbb R}`, y así bastará teclear `\R` cada vez que queramos escribir $\mathbb{R}$.

Otro ejemplo: LaTeX dispone por defecto de la instrucción `\sin`, para escribir la función seno. Si en lugar de $\sin x$ queremos escribir $\operatorname{sen} x$, sólo tenemos que definir `\sen` mediante:

```
\def\sen{\mathop{\mbox{\normalfont sen}}\nolimits}
```

Lo definimos con `\mathop` para que LaTeX lo trate como un operador matemático a efectos del espaciado, y al final ponemos `\nolimits` para que al escribir `$\sen^2x$` obtengamos siempre $\operatorname{sen}^2 x$ y no $\operatorname{sen}^2 x$.

En cambio, si queremos redefinir `\max` para que obtener máx con acento, la forma de hacerlo es

```
\def\max{\mathop{\mbox{\normalfont máx}}}
```

donde ahora no ponemos `\limits`, para que al teclear `$\max_{i\in I}a_i$` obtengamos $\max_{i\in I} a_i$ en modo texto y

$$\max_{i\in I} a_i$$

en modo `display`.

En general, los comandos de que disponemos para especificar la categoría de un signo son `\mathbin` (operador binario), `\mathop` (operador), `\mathrel` (relación), `\mathord` (ordinario), `\mathnormal` (normal). Es raro que necesitemos definir un delimitador, un acento o un signo de puntuación.

Las definiciones pueden usarse para dar un nombre manejable a instrucciones complejas. Por ejemplo, mediante

```
\def\ll{\discretionary{l-}{l}{\hbox{l$\cdot$l}}}
```

obtenemos la ele geminada que aparece en palabras catalanas como cèl·lula (que se obtiene con `cè\ll ula`, notemos la necesidad de dejar un espacio en blanco tras la `\ll`, que luego no se refleja en el texto generado). La definición es complicada porque prevé la posibilidad de que la palabra deba ser partida por las eles, en cuyo caso el punto se sustituye automáticamente por un guión.

Ya hemos usado definiciones para traducir algunas de las palabras que LaTeX escribe por defecto en inglés. La tabla 2 contiene los nombres internos de la mayoría de las palabras que LaTeX conoce junto con una posible traducción. La forma de redefinirlas es, por ejemplo,

```
\def\abstractname{Resumen}
```

Tabla 2: Palabras predefinidas en LaTeX

`\abstractname`	Resumen	`\listfigurename`	Índice de Figuras
`\alsoname`	véase también	`\listtablename`	Índice de Tablas
`\appendixname`	Apéndice	`\pagename`	Página
`\bibname`	Bibliografía	`\partname`	Parte
`\chaptername`	Capítulo	`\prefacename`	Prefacio
`\contentsname`	Índice General	`\refname`	Referencias
`\figurename`	Figura	`\seename`	véase
`\indexname`	Índice de Materias	`\tablename`	Tabla

La instrucción `\today` genera la fecha de hoy. Por ejemplo: May 20, 2019.

Si la queremos en castellano, podemos redefinirla así:

```
\def\today{\number\day~de\space\ifcase\month\or
enero\or febrero\or marzo\or abril\or mayo\or junio\or
julio\or agosto\or septiembre\or octubre\or noviembre\or diciembre\fi
\space de~\number\year}
```

de modo que ahora el resultado es: 20 de mayo de 2019.

Poniendo en el preámbulo `\usepackage[spanish]{babel}` estas traducciones se realizan de forma automática, pero así podemos personalizarlas más fácilmente.

La instrucción LaTeX para la aritmética modular deja un espacio muy grande a la izquierda de los paréntesis. Por ejemplo, si escribimos `$5\equiv 3\pmod {10}$` se obtiene $5 \equiv 3 \pmod{10}$. Si queremos corregir esta distancia y, de paso, añadirle el acento al módulo, podemos definir

```
\def\mod#1{\allowbreak\mkern5mu({\rm m\acute od}\,\,#1)}
```

con lo que ahora el efecto es $5 \equiv 3$ (mód 10).

Vemos así que las definiciones aceptan parámetros. Por ejemplo, si definimos

```
\def\expizq#1#2{{}^{#1}#2}
```

al escribir `$\expizq{10}{25}$` obtenemos ${}^{10}25$.

Cabe advertir que la instrucción `\def` es TEX, mientras que LATEX tiene otra más específica con una sintaxis distinta para los parámetros, a saber, `\newcommand`, que, entre otras cosas, nos advierte con un mensaje de error si intentamos definir algo ya definido. Pese a ello, podemos sobreescribir una definición previa usando `\renewcommand`.

Por ejemplo, con `\newcommand{\potinv}[2]{{}^{#1}#2}` conseguimos el mismo efecto de superíndice a la izquierda ${}^{10}25$, esta vez escribiendo `$\potinv{10}{25}$`.

`\newcommand` permite que el primer argumento sea opcional. Por ejemplo, si definimos

```
\newcommand{\pot}[2][\alpha]{{}^{#1}#2}
```

al escriibir `$\pot[10]{25}$` (ahora con el primer argumento opcional entre corchetes) obtenemos ${}^{10}25$, mientras que si lo omitimos y escribimos ${}^{\alpha}25$, el resultado es ${}^{\alpha}25$, donde el primer argumento ha tomado el valor por defecto `\alpha` que le hemos dado en la definición.