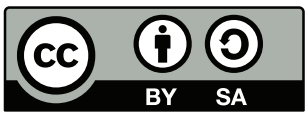
Curso express de LaTeX

Pedro Corcuera

12 de enero de 2021



Índice general

Resumen

Descripción del curso

Se ha resumido los aspectos más importantes de LaTeX. Se trata de un documento intermedio entre lo más básico y la lectura de un manual extenso.

Contenido

1. **Introducción**: donde se describe qué es LaTeX, sus ventajas e inconvenientes.
2. **Requerimientos**: sobre las herramientas necesarias para escribir un documento. Cosas sobre editores y compiladores.
3. **Un documento básico**: donde se escribe el primer documento y se aprende su sintaxis.
4. **Insertando figuras**: sobre cómo insertar imágenes en un documento, objetos flotantes, formatos y posicionamiento.
5. **Ecuaciones**: udo de los motivos por el que se utiliza extensamente LaTeX son las ecuaciones. Aquí hay una introducción sobre símbolos y comandos. También se describe de forma introductoria la gestión del espacio y las referencias cruzadas.
6. **Idioma**: sobre paquetes de idioma y codificación de entrada y de fuente. Diferencias según el compilador que se usa.
7. **Formas, tamaños y colores**: se describe el formato, en particular de cómo cambiar el tamaño, el estilo y el color del texto.
8. **La página**: sobre la posición de los elementos dentro de la página, tales como los márgenes, el interlineado y la alineación.
9. **Espacio en blanco**: sobre la gestión del espacio vertical y horizontal, así como párrafos, salto de línea y página y de cómo los trata LaTeX.
10. **Un documento científico**: se describe cuáles son las partes de un documento largo y su formato. Información sobre las unidades, el glosario y la bibliografía.
11. **Inclusión de código**: sobre resaltado de sintaxis. Mención de los paquetes `listings` y `minted`.

12. **Presentaciones**: se pueden hacer presentaciones en LaTeX. Se mencionan las diferentes opciones y de programas compatibles.
13. **Macros**: creamos nuestros propios comandos y entornos y modificamos los que ya existen.
14. **Caja de herramientas**: mencionamos las herramientas variadas que ayudan en el proceso de escritura. También los borradores, las plantillas, el control de versiones y otras herramientas externas interesantes.

Apéndices

1. **Nota sobre los archivos auxiliares**: menciona para qué sirven todos esos archivos que LaTeX genera.
2. **Enlaces de interés**: enlaces interesantes que para la redacción de estas notas.

Agradecimientos

Los agradecimientos van para Donald Knuth y Leslie Lamport quienes son los culpables del uso de LaTeX.

Capítulo 1

Introducción

1.1. Introducción a LaTeX

LaTeX es un *lenguaje de marcado*, es decir, es una manera de anotar un documento con su estructura y formato. Como cuando revisamos un texto, pensamos que tal parte debe ir en negrita y que en tal otra hay que cambiar de párrafo y escribimos unas notas en el documento para acordarnos. Un lenguaje de marcado hace esto de manera ordenada: define diferentes marcas para que luego el documento tome el formato adecuado al procesarlo. Un ejemplo es el omnipresente HTML, como su propio nombre (*HyperText Markup Language*) indica. En un navegador cuando se da al botón derecho y a *Ver código fuente de la página*, se visualiza la página con etiquetas que le indican al navegador cómo la debe mostrar.

Todos los lenguajes de marcado hacen exactamente lo mismo con la diferencia de que sus etiquetas son diferentes. Hay montones de ellos, cada uno con su campo de aplicación, por ejemplo, HTML está enfocado a desarrollar páginas web y LaTeX a escribir *documentos técnicos y científicos preferentemente* o cualquier tipo de documento.

1.2. Ventajas e inconvenientes

El motivo de usar LaTeX en la ciencia es que al haber sido inicialmente ideado por un científico[1], tuvo en cuenta las necesidades propias de los mismos. Inicialmente la principal necesidad era escribir ecuaciones decentes, hoy en día a pesar de que ese sigue siendo un punto fuerte, destacan varias cosas:

- Se trabaja en **texto plano** y se genera el documento en *pdf*/*dvi*: con ello nos podemos olvidar de las versiones del programa, la fuente siempre será accesible y siempre se podrá leer el documento final. Tiene la ventajas añadidas de que los archivos son ligeros y se puede tenerlos bajo control de versiones, algo que hay que tener en cuenta.
- Su **flexibilidad**: hay muchos paquetes hoy en día que se puede hacer de todo. Código multicolor automático, Esquemas eléctricos,

[1] Línea de tiempo: Donald Knuth desarrolló TeX y más tarde Leslie Lamport escribió un conjunto de macros para TeX a las que se llamó LaTeX.

Inconvenientes:

- Hay que **compilar**: no se visualiza en tiempo real lo que se modifica.
- **No es intuitivo**: nos enfrentamos a un texto lleno de comandos que en un primer momento nos van a parecer complicado. Usando una IDE esto mejora.
- La **documentación**: la documentación de LaTeX no está pensada para novatos.

Capítulo 2

Requerimientos

2.1. El editor y el compilador

Antes de nada hay que diferenciar dos cosas: el *editor* y el *compilador*.

Resumiendo: se puede escribir LaTeX en el Bloc de Notas si se quiere (*el editor*), luego hay que convertirlo a algo legible (*el compilador*).

Resulta que LaTeX son unas macros escritas para TeX[1] por lo que tenemos dos lenguajes de marcado que además se pueden compilar con diferentes **compiladores**[2]. Haciendo un resumen rápido:

- `tex` y `latex`: compilan respectivamente TeX y LaTeX a dvi. Para los siguientes el que solo contenga `tex` compilará TeX y el que contenga `latex` compilará LaTeX
- `pdftex`/`pdflatex`: compilan a pdf
- `xetex`/`xelatex`: compilan a pdf pero tienen la diferencia que gestionan Unicode y pueden usar las fuentes del sistema sin necesidad de configurar nada.
- `luatex`/`lualatex`: compilan a pdf. La diferencia es que están escritos en Lua.

Ahora que conocemos de compiladores veamos cómo conseguimos tener uno que nos genere los documentos. Aquí entran las **distribuciones** de LaTeX. Una distribución es un conjunto de programas y paquetes que permiten escribir sin tener que configurar todo a mano. Es decir, si instalamos una distribución tendremos los compiladores, un gestor de paquetes y otras cosas útiles. Las diferentes funcionalidades van en diferentes paquetes y se cargan solo las que interesen.

Las distribuciones más conocidas son:

- TeXLive, distribución multiplataforma, para GNU/Linux, Windows y MacOS.
- MikTeX, una distribución específica para Windows

[1] De hecho hay otras llamadas ConTeXt.
[2] También a lo que se llama "compilador" se conoce como *LaTeX engine*

La instalación de ambos es sencilla.

En cuanto los **editores**. En sí, se puede escribir en cualquier programa. Es mejor elegir uno que tenga sintaxis resaltada para que diferenciar el formato y el contenido. Se pueden dividir los editores en dos grupos:

- *Editores de propósito general*: son los que sirven para escribir en general. Van desde uno simple como gedit hasta lo más complejos Vim o Emacs. Es posible que el editor tenga un modo o un plugin que permita compilar.
- *Editores específicos* (IDE): son los editores desarrollados expresamente para escribir LaTeX. Hay varios como TeXstudio en Windows y Kile en GNU/Linux.
- *Editores online* como ejemplo (Overleaf): que directamente permite la edición y compilación desde un navegador. Es la forma más cómoda de usar Latex con la facilidad de tener una serie de plantillas y ejemplos.

2.2. Conveniencia

Como se ha mencionado la mejor opción actualmente es el uso de editores online. Uno de los mejores es Overleaf que tiene muchas ventajas.

2.3. Recapitulación:

Resumiendo, para escribir en LaTeX se necesita:

- Un **editor**, puede ser uno de uso general (como Emacs) o uno específico para LaTeX (como Kile). Si el editor no es capaz de compilar directamente es necesario una terminal.
- Una **distribución** de LaTeX, será diferente según el sistema operativo. La distribución incluirá diferentes compiladores.

2.4. Referencias

What is the difference between TeX and LaTeX? en StackExchange

LaTeX/compilation en Wikibooks

XeTeX en la wiki

LuaTeX

Why choose LuaLaTeX over XeLaTeX? en StackExchange

The differences between TeX engines en StackExchange

Choosing a LaTeX Compiler

Free TeX implementations

LaTex/Installation en Wikibooks

Comparison of TeX editors en la wiki

Capítulo 3

Un documento básico

En este capítulo se creará el primer documento, por lo que antes tenemos que describir su estructura.

3.1. Un documento y sus partes

Un documento escrito en LaTeX tiene la siguiente estructura:

Código 3.1: Ejemplo de documento escrito con LaTeX

```
\documentclass[a4paper,10pt]{article}

% PREÁMBULO

% Paquetes

\usepackage[utf8]{inputenc}
\usepackage[spanish,es-tabla]{babel}
\usepackage[T1]{fontenc}

\usepackage{listings}

% Comandos

\renewcommand{\lstlistingname}{Código}
\renewcommand{\lstlistlistingname}{Índice de fragmentos de c↩
    ódigo fuente}

% Opciones

\title{Python 2.*}
\author{Perico Palotes}

%%%%%%%%%%%%%%%%%%%%%%%%

\begin{document}
\maketitle
```

```
\begin{abstract}
Este documento es una pequeña guía de Python
\end{abstract}

\tableofcontents

\section{Sobre el lenguaje}

\begin{itemize}
    \item Interpretado
    \item Indentación obligatoria
    \item Distingue mayúsculas - minúsculas
    \item No hay declaración de variables (\textit{dynamic ↩
        typing})
    \item Orientado a objetos
    \item Garbage colector: quita los objetos a los que no ↩
        haga referencia nada
\end{itemize}

\end{document}
```

El documento tiene dos cosas a resaltar:

- Cosas que empiezan por contrabarra, los **comandos**
- Texto que va entre un `\begin` y un `\end`, los **entornos**

Los comandos son la manera en la que nos comunicamos con LaTeX y un entorno es el ámbito donde se aplica un formato.

Un entorno superimportante es el entorno `document`, que contendrá el documento. Todo lo que va entre la definición de documento (`\documentclass`) y el inicio del entorno `document` se conoce como **preámbulo** y es donde se cargan paquetes (`\usepackage`), se definen comandos y se establecen opciones.

3.2. Nuestro primer documento

El documento más básico que se puede hacer es este:

```
\documentclass{article}

\begin{document}
Hola
\end{document}
```

Se guarda en un documento documento `hola.tex` por ejemplo y se compila con el botón Recompilar (en el entorno Overleaf).

3.3. Un poco de sintaxis

Para escribir un documento un poco más interesante es necesario aprender unos pocos comandos.

Antes de nada un asunto con las opciones de idioma. Para no tener problemas con los acentos es necesario poner en el preámbulo, justo debajo de `\documentclass`:

```
\usepackage[spanish]{babel} % sustituir spanish por el ↩
    idioma
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

En el Capítulo 6 se profundizará en este tema.

3.3.1. Título, capítulos y secciones

Una cosa importante de LaTeX es que desacopla el contenido del documento de su formato. Nosotros le diremos cuál es el título del documento y dónde comienzan las secciones y Latex les dará el formato correspondiente según el tipo de documento y las opciones que se hayan establecido previamente.

Por ejemplo los tipos de documento `article` y `book` como su nombre indica, el primero se utiliza para escribir artículos y el segundo libros. Así, cuando le digamos que escriba el título, para el caso del artículo lo escribirá en la parte superior de la página con el texto debajo, pero para el caso del libro creará una portada. Para ambos casos la sintaxis es exactamente la misma:

```
\documentclass{article} % O book

% Definimos el título
\title{Título del documento}

\begin{document}
\maketitle % Creamos el título
\end{document}
```

Del mismo modo, solo es necesario especificar el título de la sección o el capítulo y Latex le dará el formato correspondiente. Otra diferencia entre las clases `article` y `book` es que `article` no tiene capítulos.

Para definir capítulos y secciones utilizamos los comandos `\chapter` y `\section` en el cuerpo del documento, es decir, después de `\begin{document}`. Por ejemplo:

```
\chapter{Capítulo numerado}
\section{Primera sección}
\subsection{Primera subsección}

\chapter*{Capítulo sin numerar}
\section*{Sección sin numerar}
```

Como se observa, se puede usar los comandos de sección y capítulo con el asterisco para que no las numere. Otra cosa interesante es el grado de anidación, tenemos secciones, subsecciones, subsubsecciones, párrafos (`\paragraph`) y subpárrafos (\subparagraph), cada uno con su formato

definido. La clase `book`[1] también tiene por encima de las secciones, capítulos y partes (`\part`). Después aprenderemos a personalizar todos estos formatos porque si hay algo bueno que tiene LaTeX es que nos deja cambiar *absolutamente todo* y con relativa facilidad.

3.3.2. Listas

LaTeX tiene dos tipos de listas: las numeradas y las sin numerar. Son respectivamente los entornos `enumerate` e `itemize`. Se usan exactamente igual, así que solo se ejemplifica uno de ellos:

```
\begin{itemize}
  \item Primer ítem
  \item Segundo ítem
\end{itemize}
```

Se puede mezclar y anidar estos dos entornos. Las viñetas y la indentación cambiarán automáticamente. Por ejemplo:

```
\begin{enumerate}
    \item Primer ítem
    \item Segundo ítem con subítems
    \begin{itemize}
        \item Ítem sin numerar
    \end{itemize}
    \item Tercer ítem
\end{enumerate}
```

Quedará así:

1. Primer ítem
2. Segundo ítem con subítems
    - Ítem sin numerar
3. Tercer ítem

3.3.3. Imágenes

Para colocar una única imagen:

- Necesitamos llamar al paquete `graphicx` en el preámbulo. Para eso escribiremos `\usepackage{graphicx}` entre `\documentclass` y `\begin{document}`
- Las imágenes se insertan con el comando `\includegraphics[opciones]{ruta}`
- Si queremos ponerles un pie de figura y una etiqueta, decidir su posición en la página y demás necesitamos el entorno `figure`

Un ejemplo de cómo insertar una imagen con en el entorno `figure`:

[1] La clase `book` no es la única que tiene partes

```
\begin{figure}[h] % opción de posicionamiento
    \caption{Pie de imagen}
    \centering % imagen centrada
    % Imagen 50% de ancho del texto
    \includegraphics[width=0.5\textwidth]{ruta_a_la_imagen}
\end{figure}
```

Una cosa importante de LaTeX son los objetos *flotantes*, es decir, los que si no les obligamos, se colocan en el lugar que mejor les venga del documento. Esto es lo que nos ocurre con las imágenes al usar el entorno `figure`. Nos ayuda a que no haya huecos en el documento pero a veces junta todas las imágenes en una misma página o al final del documento. Para evitar esto tenemos las opciones de posicionamiento, que se verá cuando profundicemos en las imágenes.

3.3.4. Tablas

Las tablas son lo peor de LaTeX. Se puede usar aplicaciones online y luego pegar el resultado.

Lo que debemos de saber de las tablas es lo siguiente:

- El contenido se especifica con el entorno `tabular`. Las celdas se separan con el ampersand y se cambia de línea con dos contrabarras.
- Si se quiere poner un pie de tabla y una etiqueta, decidir su posición en la página y demás es necesario el entorno `table`
- Si utilizamos el entorno `table` la tabla se volverá *flotante*

Un ejemplo de tabla sencilla es:

```
\begin{table}
    \begin{tabular}{|ll|}
      \hline % Separador
      Columna 1 & Columna 2 \\
      1         & 2         \\
      3         & 4         \\
      \hline
    \end{tabular}
    \caption {Pie de tabla}
\end{table}
```

Más adelante se profundizará en las tablas.

3.3.5. Ecuaciones

Las ecuaciones son la razón por la que se usa habitualmente LaTeX. Hay dos tipos de ecuaciones en LaTeX: las que van dentro de la línea y las que tienen línea propia. Las primeras van entre signos de dólar y las segundas dentro del entorno `equation`. Por ejemplo:

```
Imaginemos que $a=1$ en la siguiente ecuación:
\begin{equation}
ax^2 + 1 = 0
\end{equation}
```

Del mismo modo que ocurría con las secciones, si se utiliza el asterisco la ecuación no estará numerada.

En las ecuaciones todo se define con comandos, por ejemplo, `\frac{numerador}{denominador}` se usa para escribir una fracción y `\omega` para la letra griega omega.

Hay herramientas que nos pueden ayudar a escribir ecuaciones:

- **Editores online**: son editores con su GUI que facilitan la tarea cuando todavía no conocemos los comandos de LaTeX, por ejemplo tenemos Latex Equation Editor, LaTeX4technics, Codecogs, LaTeX Equation Editor.
- **Detexify**: una aplicación que busca el símbolo de LaTeX más parecido a algo que le pintemos.

Las ecuaciones se describen con más detalle en el Capítulo respectivo.

3.4. Referencias

LaTeX/Document Structure en Wikibooks

Environments en ShareLaTeX

Latex Standard Environments

LaTeX documentclass options illustrated

Sections and chapters en ShareLaTeX

LaTeX/Tables en Wikibooks

Inserting images en ShareLaTeX

The Comprehensive LaTeX Symbol List

Short Math Guide for LaTeX

Capítulo 4

Insertando figuras

Para insertar figuras hacen falta dos cosas:

- El paquete `graphicx`, que sustituye al paquete `graphics` y los ayuda a la hora de insertar las imágenes y compilar.
- El comando `\includegraphics[opciones]{ruta}`. Aprovechamos para aprender que los parámetros obligatorios van entre llaves y los opcionales entre corchetes.

Vamos a dividir esta entrada en dos partes: insertar una única figura e insertar una figura formada por subfiguras.

4.1. Una sola figura

Añadir una figura es simple. Solo tenemos que hacer:

```
\includegraphics{ruta_a_la_figura}
```

No hace falta ni siquiera poner la extensión, Latex buscará la imagen correspondiente. Evidentemente, esto tiene un problema: nos pondrá la imagen *ahí mismo* con su tamaño original. Lo primero aún no lo podemos solucionar, pero el tamaño se puede ajustar con las opciones.

4.1.1. Opciones

Las opciones son lo que le pasamos al comando entre los corchetes y nos permiten controlar cosas de la imagen. Las más usadas son:

- `height`: altura que debe tener la figura, escalará el gráfico hasta que tenga esta altura
- `width`: anchura que debe tener la figura, escalará el gráfico hasta que tenga esta anchura
- `scale`: cuánto hay que escalar la figura, sobre 1
- `angle`: cuánto hay que girar la figura, en grados

Por ejemplo, si queremos reducir la figura a la mitad y girarla 90 grados hacemos:

```
\includegraphics[angle=90,scale=0.5]{ruta_a_la_figura}
```

Es interesante utilizar `height` y `width` en combinación con las longitudes que define LaTeX, por ejemplo, para que una figura tenga la anchura del texto haríamos:

```
\includegraphics[width=\textwidth]{ruta_a_la_figura}
```

Podemos modificar también esta longitud, por ejemplo, para que sea un 70 % de la anchura del texto:

```
\includegraphics[width=0.7\textwidth]{ruta_a_la_figura}
```

La ventaja de este sistema es que si cambiamos los márgenes la figura se adaptará sola. Ahora vamos a ver cómo gestionamos la posición de la figura.

4.1.2. Figuras flotantes

Anteriormente se ha comentado acerca de los *objetos flotantes* y de cómo convertíamos una figura en flotante al usar el entorno `figure`. Esto permite, aparte de ponerle un pie de figura y una referencia, decidir su posición en la hoja. También tenemos la opción de rodear la imagen de texto con el entorno `wrapfigure`.

Posición

LaTeX de por sí pone las figuras donde mejor quedan (según su criterio) y nosotros le damos sugerencias de lo que preferimos. Podemos obligarle a poner las figuras en determinado lugar, pero no suele ser buena idea, es mejor reservar esta opción en casos extremos. Esta es la sintaxis:

```
\begin{figure}[posición]
  \includegraphics[opciones]{ruta}
\end{figure}
```

La opción `posición` puede tomar estos valores:

- `h` (*here*), le decimos que ponga la imagen más o menos aquí
- `t` (*top*), preferiblemente en la parte superior de la página
- `b` (*bottom*), preferiblemente en la parte inferior de la página
- `p` (*page*), que junte los objetos flotantes en una página
- `!` que ignore sus reglas internas de posicionamiento
- `H` que ponga la imagen *justo aquí*, similar a `h!` y con muchas papeletas de hacer cosas raras

Estas opciones se pueden combinar, por ejemplo, `tb` solo probará arriba y abajo. El orden no afecta. Otra cosa a tener en cuenta es la alineación de la figura. Por defecto se alinean a la izquierda, podemos cambiarla con los siguientes comandos:

- `\centering`: para centrar
- `\raggedleft`: para alinear a la derecha
- `\raggedright`: para alinear a la izquierda

que pondríamos dentro del entorno `figure` antes de insertar la imagen:

```
\begin{figure}[posición]
  \centering
  \includegraphics[opciones]{ruta}
\end{figure}
```

Pie de figura y referencia

Como hemos dicho, el entorno `figure` nos permite también poner un pie de figura y una etiqueta a la figura:

```
\begin{figure}[posición]
  \includegraphics[opciones]{ruta}
  \caption{Pie de figura}
  \label{etiqueta}
\end{figure}
```

La etiqueta sirve para hacer referencia a la figura en el texto con el comando `\ref{etiqueta}`, por ejemplo:

```
\begin{figure}[h]
  \includegraphics[scale=0.7]{Figuras/gatos}
  \caption{Unos gatos guapos}
  \label{fig:gatos}
\end{figure}

Como se ve en la Figura \ref{fig:gatos}, hay gatos negros y ↩
    blancos
```

LaTeX nos numerará las figuras correctamente de forma automática y citará la figura correspondiente cuando se haga una referencia. Una idea inteligente es etiquetar las cosas de manera que luego evitar confusiones porque no se sabe si una determinada etiqueta hace referencia a una ecuación, a una tabla o a un capítulo. Una opción podría ser:

- `fig:` para las figuras
- `eq:` para las ecuaciones
- `sec:` para las secciones

Hay que procurar ser coherentes.

4.2. Sobre formatos

Una última nota antes de tratar múltiples figuras. Cuando mencionamos los compiladores con los que LaTeX se puede compilar a *dvi* y *pdf* dependiendo de si usamos `latex` o `pdflatex` (o las otras opciones con y sin `la`). Para las imágenes esto es importante: `latex` solo compilará si las imágenes están en formato *eps*; `pdflatex`, en cambio, acepta *pdf*, *png* y

jpg. El caso del formato *eps* al compilar a *pdf* es especial, técnicamente no se inserta la imagen en *eps*, sino que por detrás se llama al programa `epstopdf`, se convierte a *pdf* y se inserta. En general se hace solo, pero depende del programa. Una cosa interesante de `epstopdf` es que se puede usar aparte en la terminal, por ejemplo:

```
epstopdf archivo.eps
```

creará `archivo.pdf`.

4.3. La fiesta de las figuras

Veamos cómo hay que hacer si queremos incluir una figura formada por subfiguras, es decir, que un grupo de figuras que van juntas y comparten pie de figura. Para esto se usaba primero el paquete `subfigure`, luego `subfig` y ahora se supone que debe usarse `subcaption`. Resulta que `subfig` tiene problemas con `hyperref` (el paquete de hacer hipervínculos) y `subcaption` no. Además, `subcaption` no es compatible con ninguno de los dos anteriores. El paquete `subcaption` define el entorno `subfigure`, que se usa a su vez dentro del entorno `figure`. Tiene esta especificación:

```
\begin{subfigure}[posición]{ancho}
  % Contenido
\end{subfigure}
```

Dentro del entorno `subfigure` se escribe exactamente lo mismo que dentro del entorno `figure`. Por ejemplo, si quisiéramos poner dos imágenes en paralelo haríamos algo así:

```
\documentclass{article}
% Necesitamos los paquetes graphicx, caption y subcaption
\usepackage{graphicx}
\usepackage{caption}
\usepackage{subcaption}
\begin{document}
  \begin{figure}
    \centering
    \begin{subfigure}[h]{0.45\textwidth}
      \includegraphics\[width=\textwidth\]{figura1}
      \caption{Pie de figura1}
      \label{fig:figura1}
    \end{subfigure}
    \begin{subfigure}[h]{0.45\textwidth}
      \includegraphics[width=\textwidth]{figura2}
      \caption{Pie de figura2}
      \label{fig:figura2}
    \end{subfigure}
    \caption{Pie general}
  \end{figure}
\end{document}
```

Se observa que no hay separación entre el fin del primer entorno `subfigure` y el inicio del segundo, si se deja una línea en blanco, la siguiente figura en lugar de ponerla en paralelo la pondrá debajo. Ahí en medio también se puede indicar cómo de separadas queremos las

figuras, pero para es necesario aprender primero cómo funciona el espacio blanco en LaTeX y ese tema se estudia en otro capítulo.

4.4. Referencias

Packages in the graphics bundle (pdf)

LaTeX/Importing Graphics en Wikibooks

Which graphics formats can be included in documents processed by latex or pdflatex? en StackExchange

Inserting Images en ShareLaTeX

How to influence the position of float environments like figure and table in LaTeX? en StackExchange

subcaption vs.~subfig: Best package for referencing a subfigure en StackExchange

Capítulo 5

Las ecuaciones

Como se mencionó anteriormente, hay dos tipos de ecuaciones en LaTeX:

- las que van **dentro de una línea**, que se escriben entre signos de dólar y se suelen conocer como *inline*
- las que tienen **línea propia**, que usan el entorno `equation` o el atajo `\[...\]`

Aquí tenemos un ejemplo usando los dos tipos:

```
\begin{equation}

  e^{i\pi} + 1 = 0

\end{equation}

donde $i =\sqrt{-1}$
```

Como se observa, las ecuaciones se escriben mediante comandos, algo que inicialmente inicialmente resulta complicado pero cuando se tiene un poco de práctica, es muy eficaz. Cuando se usa un editor específico, hay ayudas como una barra con los símbolos más usados. Al escribir ecuaciones es recomendable cargar los siguientes paquetes:

- `amsmath` (AMS Math), que mejora el comportamiento y el aspecto de las ecuaciones. Permite, por ejemplo, añadir un asterisco en el entorno `equation` para crear ecuaciones sin numerar.
- `amsthm` (AMS Theorem), que define los entornos teorema y demostración.
- `amssymb` (AMS Symbol), que carga a su vez `amsfonts` e incluye una colección de símbolos matemáticos.

Se puede cargar todos a la vez añadiendo la siguiente línea al *preámbulo*[1]:

```
\usepackage{amsmath, amsthm, amssymb}
```

[1] Recordemos: el *preámbulo* es lo que hay entre la definición del documento (`\documentclass`) y el inicio del entorno `document` (`\begin{document}`).

El paquete AMS que precede a todos ellos viene de *American Mathematical Society*, los que originalmente desarrollaron estos paquetes.

5.1. Comandos

Antes de ver más sobre la sintaxis de ecuaciones, se listan unas herramientas interesantes sobre todo para los novatos:

- **Editores de ecuaciones online**: hasta coger práctica con las ecuaciones, aparte de la barra del IDE, existen editores online como texquationeditor o codecogs o tutorialspoint o Online LaTeX Equation Editor.
- **Detexify**: si no se sabe cómo se llama un símbolo y, por lo tanto, no podemos buscar su comando tenemos Detexify, una aplicación en el que se pinta el símbolo que se desea buscar y produce los más parecidos.

5.1.1. Símbolos comunes

Hay muchos símbolos, a continuación se escriben unos pocos comandos.

- *Sumas, restas y exponenciales*: se hacen con el símbolo de toda la vida `+`, `-` y `^`
- *Multiplicaciones*: hay variedad según los gustos, si queremos el punto usamos el comando `\cdot` el aspa usamos `\times`. Es mejor no usar la equis ni el asterisco.
- *Raíces*: se hacen con el comando `\sqrt{argumento}` si son raíces cuadradas y añadiendo el número como argumento opcional (entre corchetes) para cualquier otra raíz. `\sqrt[raíz]{argumento}`
- *Integrales*: se usa el comando `\int`, si tienen límites definidos se escribe `\int_{inferior}^{superior}`. Por ejemplo, esta integral impropia $\int_0^\infty$ se conseguiría así `\int_{0}^{\infty}`. Observar que las integrales, a diferencia de las raíces, no llevan llaves. Esto ocurre porque la raíz necesita saber cómo de largo es el contenido, la integral es simplemente el símbolo.
- *Sumatorios*: son como las integrales pero con el comando `\sum`
- *Fracciones*: tan sencillas como `\frac{numerador}{denominador}`

En las referencias hay listas de símbolos.

5.1.2. Letras griegas

Una de las mejores cosas de LaTeX es su método para escribir letras griegas, tan sencillo como escribir su nombre en minúsculas para la letra en minúscula y ponerle la primera en mayúscula para una letra griega en mayúscula. Por ejemplo:

> `\omega` crea ω (*omega minúscula*) y `\Omega` crea a su vez Ω (*omega mayúscula*)

5.1.3. Operadores

Los operadores son las funciones cuyo nombre se escribe en texto, como *sin* o *ln*. LaTeX tiene algunos de ellos definidos y es importante usarlos para que las ecuaciones queden bien. Por ejemplo:

```
\sin^2 x + \cos^2 x = 1
```

Que crea:

$$\sin^2 x + \cos^2 x = 1$$

5.1.4. Matrices

Funcionan de manera similar a las tablas (las columnas se separan con el ampersand y se salta de línea con `\\`), pero usan el entorno `matrix` y relacionados. El entorno `matrix` crea una matriz sin delimitadores. Los entornos `pmatrix`, `vmatrix`, `Vmatrix` `bmatrix` y `Bmatrix` añaden respectivamente paréntesis, barras[2], barras[3] dobles, corchetes y llaves. Estos entornos centran el contenido, si se quiere cambiar la alineación hay que usar sus variantes con asterisco y darle un argumento. Ejemplo de una matriz sencilla:

```
\begin{equation}
  \begin{matrix}
    a & b & c \\
    d & e & f \\
    g & h & i \\
  \end{matrix}
\end{equation}
```

Sobre los paréntesis

Si no se quiere memorizar que el `pmatrix` pone un paréntesis y el `vmatrix` otro delimitador, se puede poner los delimitadores a discreción y usar siempre el entorno `matrix` pero hay que tener en cuenta que en LaTeX hay dos tipos de paréntesis: los de tamaño fijo y los que se adaptan al contenido. Los de tamaño fijo son tal cual el símbolo según el teclado, los adaptativos son comandos formados por `\left` o `\right`, según el lado, más el símbolo. Por ejemplo, `\left(` crea el paréntesis adaptativo del lado izquierdo, `\right]` el corchete adaptativo de la derecha y así con todos. Los únicos un poco diferentes son los comandos para las llaves, que requieren una barra de escape y son respectivamente `\left\{` y `\right\}` Por ejemplo, para encerrar la matriz anterior con corchetes se hace lo siguiente:

```
\begin{equation}
  \left[
  \begin{matrix}
    a & b & c \\
    d & e & f \\
    g & h & i \\
```

[2] Como las de un determinante
[3] Como las de una norma

```
  \end{matrix}
  \right]
\end{equation}
```

5.2. Gestión del espacio

Al igual que con el texto, LaTeX gestiona el espacio entre los símbolos de forma automática. En general lo mejor es dejarle hacer, pero hay ocasiones que no quedan bien y hay que hacerlo manualmente. Para ello se puede utilizar útiles, aunque hay muchos más, que no son específicos de las ecuaciones pero es donde suelen resultar más necesarios:

- `\,`: genera un espacio en blanco estrecho
- `~`: crea un *espacio duro*, es decir, un espacio que impide que se salte de línea en medio.

En las referencias hay más detalles sobre el tema.

5.3. Referencias cruzadas

Igual que las imágenes, las ecuaciones también se pueden referenciar haciendo uso de los comandos `\label` y `\ref`. El primero de ellos permite darle un nombre identificativo a una ecuación y el segundo la cita. Al igual que las figuras, para añadir una etiqueta a una ecuación es necesario utilizar el entorno `equation`, no sirve para las ecuaciones *inline*. Por ejemplo para citar la ecuación del primer ejemplo:

```
\begin{equation}
  e^{i\pi} + 1 = 0
  \label{eq:euler}
\end{equation}

Como vemos en la Ecuación \ref{eq:euler}
```

Que produce este resultado:

$$e^{i\pi} + 1 = 0$$

Como vemos en la Ecuación 1

Añadir `eq:` a la etiqueta no es necesario pero facilita el trabajo al no tener las etiquetas para las figuras, las secciones y demás elementos mezclados. También se puede definir un comando para que añada la palabra *Ecuación* al número. Para ello se crea el comando:

```
% Estructura \newcommand{\nombre}[nroargs]{Descripción}
\newcommand{\refeq}[1]{Ecuación~\ref{#1}}
```

Esto mismo se consigue con el comando `\autoref` del paquete `hyperref` con la ventaja de que nos pone la palabra correcta en todos los casos, ya sean tablas, figuras o ecuaciones sin necesidad de definir un comando para cada uno.

5.4. Grupos de ecuaciones

Otro tema interesante es poder escribir un grupo de ecuaciones que comparta la misma etiqueta. Esto es posible gracias a diferentes entornos como: `align` del paquete `amsmath`. Permite crear un sistema de ecuaciones que alineará según el símbolo que marquemos con un ampersand. Por ejemplo, las 3 leyes de la termodinámica quedarían así, alineadas según el símbolo de igual:

```
\begin{align}
  \Delta U &= Q -W \\
  \delta S &= T \mathrm{d}S \\
  S &=\mathrm{k_B}\ln\Omega
\end{align}
```

$$\begin{aligned}
\Delta U &= Q - W \\
\delta S &= T\mathrm{d}S \\
S &= \mathrm{k_B}\ln\Omega
\end{aligned}$$

Al igual que con el entorno `equation`, con `align` también es posible añadir una etiqueta o usar el asterisco para que no numere la ecuación.

5.5. Formato

LaTeX permite adaptar el formato de las ecuaciones a nuestros gustos o exigencias externas (por ejemplo formato de revistas científicas, normas ISO . . .). Un formato típico es el siguiente:

- *Cursiva para las variables*: LaTeX lo hace por defecto
- *Operadores y constantes rectos*: para los operadores del propio LaTeX como `\sin` o `\log` no se hace nada, lo hace de forma automática. Para el resto hay dos opciones: usar `\mathrm` o definirlos como operadores en el preámbulo con `\DeclareMathOperator` del paquete `amsmath`. La última forma se describirá más adelante, aunque va un adelanto:

```
\usepackage{amsmath}
\DeclareMathOperator{\comando}{descripción}
```

- *Matrices y vectores en negrita*: para ello usaremos `\mathbf` para las letras y `\boldsymbol` para los símbolos o letras griegas.

Un ejemplo con todos ellos podría ser la definición de la matriz de rigidez para el método de los elementos finitos:

```
\begin{equation}
  \mathbf{K}=\int_V \mathbf{B}^\intercal \mathbf{D B}\mathrm{d}x\mathrm{d}y \mathrm{d}z
\end{equation}
```

que quedaría como:

$$\mathbf{K} = \int_V \mathbf{B}^\intercal \mathbf{D} \mathbf{B} \mathrm{d}x \mathrm{d}y \mathrm{d}z$$

5.6. Referencias

LaTeX/Mathematics en WikiBooks

List of Greek letters and math symbols en ShareLaTeX

Matrix environments en LaTeX Wiki

Brackets and Parentheses en ShareLaTeX

Operators en ShareLaTeX

What commands are there for horizontal spacing? en StackExchange

Lista de símbolos matemáticos (pdf)

What does each AMS package do? en StackExchange

How to typeset equations in LaTeX (pdf)

Capítulo 6

El idioma

Si no se escribe en inglés, es necesario configurar LaTeX para que se adapte a nuestro idioma. Para ello, es necesario elegir un paquete de idioma y una codificación.

6.1. El paquete de idioma

El paquete de idioma realiza dos funciones principales:

- Gestiona que las palabras claves como Capítulo o la fecha se escriban en el idioma elegido.
- Aplica la silabación[1] correcta para el idioma concreto, así como otras reglas tipográficas.

Dependiendo del compilador usado se tendrá que usar un paquete de idioma u otro:

- Si se usa usa `pdflatex`, es necesario cargar el paquete `babel` con el idioma elegido como opción, por ejemplo:

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

- Si se usan compiladores más modernos como `xelatex` o `lualatex`, se puede usar el paquete `polyglossia`[2]. Este paquete es más simple y ligero aprovechando que estos compiladores ya gestionan la codificación y las fuentes de por sí. En este caso no se carga el idioma como opción, sino que se selecciona con el comando `\setmainlanguage`:

```
\usepackage{polyglossia}
\setmainlanguage{spanish}
```

[1] partición de una palabra cuando no cabe en la línea
[2] exceptuando algún caso especial, `babel` también funciona.

6.1.1. Redefinir palabras clave

Puede pasar que no sea de nuestro agrado la palabra que el paquete de idioma utiliza para referirse a cierta parte del documento, por ejemplo, utilizando la opción de español LaTeX llamará *Cuadro* a las tablas. Si es así, podemos redefinir el nombre del elemento. Es sencillo, se usa `\renewcommand` con la siguiente estructura:

```
\renewcommand{Comando a redefinir}{Nueva definición}
```

Hay diferencias si se usa `babel` o `polyglossia`.

El caso babel

Si se usa `babel`, el comando que redefinimos tendrá un nombre formado por el nombre del idioma y el nombre de la palabra clave. Por ejemplo, para cambiar Cuadro por Tabla en español haríamos:

```
\renewcommand{\spanishtablename}{Tabla}
```

Del mismo modo, si se quiere modificar cómo llama LaTeX a las figuras en francés, se redefine el comando `\frenchfigurename` y así con todos.

Cuadro 1: Traducciones

`\refname`	`\spanishrefname`	Referencias
`\abstractname`	`\spanishabstractname`	Resumen
`\bibname`	`\spanishbibname`	Bibliografía
`\chaptername`	`\spanishchaptername`	Capítulo
`\appendixname`	`\spanishappendixname`	Apéndice
`\contentsname`	`\spanishcontentsname`	índice general [a]
`\listfigurename`	`\spanishlistfigurename`	índice de figuras
`\listtablename`	`\spanishlisttablename`	índice de cuadros
`\indexname`	`\spanishindexname`	índice alfabético
`\figurename`	`\spanishfigurename`	Figura
`\tablename`	`\spanishtablename`	Cuadro
`\partname`	`\spanishpartname`	Parte
`\enclname`	`\spanishenclname`	Adjunto
`\ccname`	`\spanishccname`	Copia a
`\headtoname`	`\spanishheadtoname`	A
`\pagename`	`\spanishpagename`	página
`\seename`	`\spanishseename`	véase
`\alsoname`	`\spanishalsoname`	véase también
`\proofname`	`\spanishproofname`	Demostración

[a] Pero sólo «índice» en article.

Figura 6.1: Tabla que indica cómo llama a cada uno de los elementos. Del manual del estilo `spanish` del paquete `babel`

En el caso concreto de las tablas y para `pdflatex` se puede cargar el paquete `babel` con la opción `es-tabla` y evitamos problemas, pero para el resto de elementos hay que hacer lo mencionado.

El caso polyglossia

Si se usa `polyglossia`, en su manual indica que hay que cambiar el nombre de la palabra clave con `\gappto` (o `\addto` para que sea compatible con `babel`)[3]. Por ejemplo:

```
\gappto\captionsspanish{\renewcommand{\tablename}{Tabla}}
```

6.1.2. Texto en diferentes idiomas

Aquí se indica cómo se cambia de idioma en un mismo documento.

El caso `babel`

En este caso simplemente se carga como opciones del paquete `babel` todos los idiomas que se va a usar en el documento:

```
\usepackage[idioma1, idioma2]{babel}
```

Los activamos en el texto con `\selectlanguage`:

```
\usepackage[spanish, english]{babel}
\selectlanguage{spanish}

\begin{document}

  \section{Sección en español}

  \selectlanguage{english}
  \section{Section in english}

\end{document}
```

El caso `polyglossia`

Si se usa `polyglossia`, se carga el paquete y se establecen los idiomas en el preámbulo de la siguiente manera:

```
\usepackage{polyglossia}
\setmainlanguage[Opciones]{Idioma} % Idioma principal
\setotherlanguage[Opciones]{Idioma} % Otro idioma
```

Si se quiere cambiar el idioma de un trozo pequeño de texto usamos el comando `\textIDIOMA`. Por ejemplo, para poner la fecha en español:

```
\textspanish{\today}
```

Si se va a escribir un trozo de texto largo en otro idioma, es preferible usar el entorno correspondiente al idioma;

```
\begin{spanish}
  Texto en español
\end{spanish}
```

[3] En el manual de español de `babel` indica que es mejor usar directamente renewcommand y spanishtablename o el nombre que corresponda.

6.2. La codificación

La otra parte importante a la hora de configurar el idioma en LaTeX es la codificación. Hay dos tipos de codificación: la codificación de entrada y la codificación de fuente.

Configurar la codificación de entrada correctamente para un idioma (en nuestro caso español) permite escribir directamente los caracteres especiales desde el teclado, por ejemplo, poder poner `á` en lugar de `\'a`.

Una codificación de fuente correcta, por su parte, sirve para que LaTeX pueda partir las palabras donde debe y para poder buscar en el pdf resultante palabras con caracteres especiales. Si se usa una codificación de fuente inadecuada ocurren cosas curiosas como que no se pueda copiar de un pdf palabras que contengan *fi*. Esto se debe a que LaTeX trata *fi* como un único carácter, ya que es una ligadura tipográfica.

Aquí ocurre exactamente lo mismo que antes, dependiendo del compilador se tiene que usar o no diferentes paquetes.

6.2.1. El caso de pdflatex

El compilador `pdflatex` es antiguo y requiere ayuda para gestionar la codificación. Hacen falta los siguientes paquetes:

- `inputenc`: gestiona la codificación de entrada. Como opción indicamos la codificación, para ahorrar problemas usamos UTF 8
- `fontenc`: gestiona la codificación de fuente. Para escribir en castellano, se usa la codificación T1 que contiene los caracteres necesarios de los idiomas que usan alfabeto latino con acentos. Si se quiere escribir en cirílico se tiene que usar que usar la T2A.
- `lmodern` o `cm-super`: la fuente que LaTeX usa por defecto, Computer Modern, no es compatible con la codificación T1 así que se necesita usar su variante Latin Modern (paquete `lmodern`) o la Computer Modern Super (paquete `cm-super`). La experiencia indica que la Latin Modern es mejor para los acentos y la CM Super para el cirílico.

Si se quiere escribir en español, la parte del idioma en el preámbulo para `pdflatex` queda por lo tanto así:

```
\usepackage[spanish]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

Para cualquier otro idioma con acentos o caracteres especiales se necesitan los mismos paquetes y solo se requiere cambiar la opción de idioma en el paquete `babel`.

6.2.2. Xelatex y otros compiladores modernos

Los compiladores modernos gestionan ellos solos la codificación de entrada y la de fuente así que no es necesario añadir ningún paquete extra.

6.3. Resumen

Recapitulando para ver qué se requiere en cada caso.

Para compilar con `pdflatex` se añade en el preámbulo:

```
\usepackage[spanish,es-tabla]{babel} % Carga es-tabla para ↩
    Tabla en lugar de Cuadro
\usepackage[utf8]{inputenc} % Codificación de entrada
\usepackage[T1]{fontenc} % Codificación de fuente
\usepackage{lmodern} % Fuente compatible
```

Para compilar con `xelatex` se añade al preámbulo:

```
\usepackage{polyglossia}
\setmainlanguage{spanish}
% Para Tabla en lugar de Cuadro
\gappto\captionsspanish{\renewcommand{\tablename}{Tabla}}
```

Se puede utilizar el paquete `ifxetex` (que comprueba si se compila con `xelatex`), añadiendo este trozo al preámbulo y asegurar que funciona con los dos compiladores:

```
% Idioma
\ifxetex
  \usepackage{polyglossia}
  \setmainlanguage{spanish}

  % Tabla en lugar de cuadro
  \gappto\captionsspanish{\renewcommand{\tablename}{Tabla}
          \renewcommand{\listtablename}{Índice de tablas}}

\else
  \usepackage[spanish,es-tabla]{babel}
  % Para los acentos (xelatex no lo necesita)
  \usepackage[utf8]{inputenc}
  \usepackage[T1]{fontenc}
  \usepackage{lmodern}
\fi
```

Se ha añadido la parte de cambiar *Cuadro* por *Tabla* pero no es necesario.

6.4. Referencias

Manual del paquete `babel` (pdf)

Estilo spanish para el sistema babel (pdf)

Manual del paquete `polyglossia` (pdf)

LaTeX font encodings

Capítulo 7

Formas, tamaños y colores

Este capítulo trata sobre el formato del texto. Hasta ahora el texto escrito aparece con el formato por defecto, es momento de aprender a cambiar el tamaño y estilo de la letra y a escribir en diferentes colores. Para ello tenemos que entender qué es una clase y cómo afecta a nuestro documento.

7.1. Clases

El formato de un documento está definido por su clase, que especifica, entre otras cosas, cómo de grande tienen que ser los títulos, cuánto espacio tiene que haber entre los ítems de una lista o el tamaño de los márgenes. Estas clases pueden ser las típicas `article` o `book`, unas que vivan en un paquete como las clases Tufte o incluso las escritas por nosotros. Son simplemente un archivo *cls* lleno de definiciones.

La clase del documento se establece en la definición inicial:

```
% Usar la clase memoir
\documentclass{memoir}
```

Ahora que conocemos que el estilo del documento lo decide la clase, veremos cómo cambiar alguna cosa específica en el documento. Los cosas generales, por ejemplo, que todos los títulos de sección estén en cursiva es preferible redefinir la orden en cuestión.

7.2. Formato de texto

Empezamos modificando el texto. Cambiar su tamaño, aplicar diferentes estilos y una pequeña introducción al color en LaTeX.

7.2.1. Tamaño

La manera en la que LaTeX trata el tamaño de letra es bastante original: coge como referencia el tamaño de la letra del cuerpo del documento y define los demás tamaños de manera relativa respecto a este. Así, si se cambia el tamaño de letra del cuerpo todo lo demás cambia en consonancia.

Recordando que el tamaño de letra del cuerpo se puede establecer como argumento opcional en `\documentclass`:

```
\documentclass[11pt]{article}
```

Si no se indica nada, usará 10pt por defecto.

En esta tabla del libro sobre LaTeX en Wikibooks están los comandos para agrandar y reducir la letra y sus respectivos tamaños de letra según el tipo de documento seleccionado. Tanto `article` como `book` son parte de las clases estándar.

Absolute Point Sizes

size	standard classes (except *slides*), beamer			AMS classes, *memoir*			*slides*
	[10pt]	[11pt]	[12pt]	[10pt]	[11pt]	[12pt]	
\tiny	5	6	6	6	7	8	13.82
\scriptsize	7	8	8	7	8	9	16.59
\footnotesize	8	9	10	8	9	10	16.59
\small	9	10	10.95	9	10	10.95	16.59
\normalsize	10	10.95	12	10	10.95	12	19.907
\large	12	12	14.4	10.95	12	14.4	23.89
\Large	14.4	14.4	17.28	12	14.4	17.28	28.66
\LARGE	17.28	17.28	20.74	14.4	17.28	20.74	34.4
\huge	20.74	20.74	24.88	17.28	20.74	24.88	41.28
\Huge	24.88	24.88	24.88	20.74	24.88	24.88	41.28

Para usar cualquiera de estos comandos hay dos opciones, usarlos dentro de una línea o como entorno. Si se usa en la propia línea, harán efecto hasta el final del *grupo* (una zona delimitada entre llaves) o *entorno* actual. Si la zona de aplicación no está delimitada, el tamaño de letra se mantendrá hasta encontrarse con otro comando de tamaño o hasta el final del documento. Por ejemplo:

```
Texto normal {\Huge Texto gigantesco}

\normalsize Texto normal hasta nueva orden

\section{\Huge Título de sección enorme}

Texto normal

\begin{quote}
  \Large Letra grande solo para la cita
\end{quote}
```

```
Texto normal de nuevo

\small Texto pequeño hasta el final del documento
```

Si el trozo de texto que se quiere personalizar es muy largo, es mejor usar el entorno para ganar en legibilidad:

```
\begin{tiny}
    Miniletrilla
\end{tiny}
```

También se puede establecer otros tamaños de letra como se hace en los editores de toda la vida. Para ello se utiliza el comando `\fontsize`:

```
\fontsize{tamaño}{baseline skip}\selectfont Texto
```

Se puede facilitar ambos argumentos en diferentes unidades, si no se pone más que el número, LaTeX entenderá que se indica *puntos*. El segundo argumento es importante, ya que fija la distancia `\baselineskip`, es decir, la distancia entre las partes inferiores de dos líneas sucesivas. Suele ser 1.2 veces el tamaño de fuente[1].

Si se utiliza este sistema es preferible que usar una *fuente vectorial*, así estará disponible en cualquier tamaño[2]. Por ejemplo, con Computer Modern es probable que salga un aviso de este estilo:

```
LaTeX Font Warning: Font shape `OT1/cmr/m/n' in size <20> ↩
    not available
(Font)              size <20.74> substituted on input line ↩
    25.
```

Esto ocurre porque la fuente Computer Modern es del tipo *bitmap*, es decir, las letras están formadas por pixels. Esto implica que hay versiones de la fuente para tamaños concretos, no para todos. En este caso, lo más fácil es utilizar Latin Modern (`\usepackage{lmodern}`) que es igual y como es OpenType (y por lo tanto vectorial) no tiene este problema. Además es compatible con la codificación T1, algo que es necesario si se usan tildes.

7.2.2. Forma

Por el momento se ignora el tipo de fuente y se especifica las negritas, cursivas y demás cosas.

En LaTeX el estilo de cada letra pertenece a una *familia*, una *serie* y una *forma*:

- La familia indica cómo es la fuente: serif o *roman* (`\rmfamily`), sans serif (`\sffamily`) o monoespaciada (`\ttfamily`)
- La serie hace referencia al grosor del trazo: fino (`\lfseries`), medio (`\mdseries`) o grueso (`\bfseries`)
- La formas como su nombre indica describen la forma de las letras: recta (`\upshape`), cursiva (`\itshape`), `oblicua (`*') o versalita (''*).

[1] Cambiar estas cosas sin dominar el tema puede ser desastroso.
[2] Para una explicación mejor sobre los tipos de fuente ver el artículo de la Wikipedia.

La letra por defecto es serif, es de grosor medio y recta. En LaTeX se denomina `\normalfont`. Se puede hacer otras combinaciones siempre con un elemento de cada grupo. Saber esto es especialmente útil cuando se define un estilo propio.

Se puede activar diferentes familias, series y formas con diferentes comandos[ˆcomand]:

- `\textbf{}` pone el texto en negrita
- `\textit{}` sirve para activar la cursiva
- `\texttt{}` escribe en letra de máquina de escribir o monoespaciada
- `\textsc{}` para las versalitas
- `\textnormal{}` vuelve a `\normalfont`

Se pueden anidar estos comandos y escribir en letra monoespaciada, cursiva y negrita simultáneamente si deseamos. En cualquier caso, este es un tema complejo habiéndose mostrado una introducción. En las referencias se encuentra más información.

No aparece la opción de subrayar porque es una mala decisión tipográfica que proviene de los tiempos en los que se escribía a máquina. No obstante si se quiere subrayarse usa el comando `\underline{}`. También el paquete `ulem`, proporciona el comando `\ul` para subrayar.

Algo interesante es el comando `\emph{}` que sirve para enfatizar. En general se comporta como la cursiva aunque con un par de diferencias:

- Depende del contexto, por ejemplo, si se enfatiza un trozo que estaba ya en cursiva se pone recto:

```
\textit{Texto en cursiva \emph{texto recto}}
\emph{Texto en cursiva \emph{texto recto}}
```

- Se puede modificar para enfatizar según el gusto, ya sea en negrita, con colores o lo que sea. Es precisamente lo que hace el paquete `ulem` (*underline emphasis*)

7.2.3. Color

Para usar colores en LaTeX es necesario un paquete: `xcolor`, la evolución del paquete `color`. Se carga en el preámbulo con unas pocas opciones para facilitar su uso:

```
\usepackage[usenames,dvipsnames,svgnames,table]{xcolor}
```

- `usenames` permite llamar a los 16 colores básicos por su nombre en lugar de por su definición.
- `dvipsnames` da acceso a otros 64 colores extra.
- `svgnames` añade otros 150 colores más.
- `table` deja usar colores en las tablas.

Una explicación más detallada sobre los colores disponibles se ecuentra en el apartado 2.4 del manual de `xcolor`.

Para cambiar el color del texto se puede usar tanto `\textcolor` como `\color`:

```
\textcolor{red}{texto de color rojo}

{\color{magenta} texto en color rosa raro}
```

Como ocurría con los comandos para cambiar el tamaño de letra, en este caso `color` afecta hasta el final del grupo.

También se puede definir colores propios con `\definecolor` especificando un nombre o directamente cuando se vaya a usar. Para ello es necesario elegir un modelo de color (rgb, cmyk, HTML, ...) y darle el valor correspondiente:

```
% RGB
\definecolor{azulito}{rgb}{0.36, 0.54, 0.66}

% HTML (en hexadecimal y mayúsculas)
\definecolor{otroColor}{HTML}{4070A0}

% Definir directamente al usarlo
{\color[HTML]{4070A0} texto en color}
```

Definir nuestros propios colores es especialmente útil cuando se desea combinar los colores del texto con alguno específico de una imagen o si las opciones `dvipsnames` y `svgnames` no funcionan por algún motivo. En la página *LaTeX color* hay muchas definiciones de colores para seleccionar.

7.3. Resumen

Recapitulando:

- **Tamaño**: los tamaños en LaTeX son relativos y se cambian con comandos que van de `\tiny` para la letra más pequeña a `\Huge` para la más grande. También se puede establecer un tamaño de fuente concreto.
- **Forma**: los estilos de las letras se dividen en familias, series y formas que pueden combinarse entre sí. A cada elemento le corresponde un comando que activa un formato específico. El texto por defecto es serif, de grosor medio y recto, se conoce como `\normalfont`.
- **Color**: para usar colores en LaTeX es necesario el paquete `xcolor`. Sus diferentes opciones dan acceso a un número de colores predefinidos pero también existe la posibilidad de definir colores propios con `\definecolor{modelo}{definición}`. Aplicamos al texto cualquiera de estos colores con `\textcolor{color}{texto}` o con `\color{color}`.

7.4. Referencias

Classes en *The TeX catalogue*

What are the available "documentclass" types and their uses? en StackExchange

Tufte-LaTeX en CTAN

Line breaks and blank spaces

What commands are there for horizontal spacing? en TexExchange

LaTeX/Fonts en Wikibooks

Font sizes, families, and styles

Font selection in LaTeX: The most frequently asked questions (pdf)

LaTeX font commands

The `fontspec` *package*

LaTeX/Page Layout en WikiBooks

Capítulo 8

La página

Este capítulo trata del formato de la página: cómo configurar la alineación, el interlineado, los márgenes y la sangría. Un *adelanto*:

> There is a LaTeX package for it

8.1. Alineación

LaTeX justifica el texto por defecto. Para cambiar la alineación hay comandos y entornos equivalentes, los comandos hacen efecto en el grupo actual y los entornos en su contenido. Tanto los unos como los otros valen tanto para el texto como para las tablas y figuras:

- Para **alinear a la izquierda** hay el entorno `flushleft` (*nivelado a la izquierda*) y el comando `\raggedright` (*desigual a la derecha*)
- Para **alinear a la derecha** se puede elegir entre el entorno `flushright` (*nivelado a la derecha*) y el comando `\raggedleft` (*desigual a la izquierda*)
- Para **centrar** se puede usar el entorno `center` o el comando `\centering`

Hay dos diferencias entre usar un entorno y un comando para alinear:

- El entorno añade espacio blanco adicional delante y detrás de su zona de aplicación
- El comando necesita un comando que le diga dónde acaba el párrafo, es por ello que se usa dentro de entornos como `figure` o `quote`. En un caso general se tiene que añadir una línea en blanco o el comando `\par` al final del párrafo.

Los comandos son bastante prácticos cuando ya estamos dentro de un entorno porque evita escribir otro `begin` y `end` y, además, se mejora la legibilidad.

Ejemplos de texto alineado a la derecha:

```
% Texto alineado a la derecha
```

```
% Opción 1: entorno flushright
\begin{flushright}
  Texto
\end{flushright}

% Opción 2: \raggedleft + \par
{\raggedleft Texto\par}

% Opción 3: \raggedleft + línea en blanco
{\raggedleft Texto

}

% Opción 4: \raggedleft + entorno que indica final de pá↩
    rrafo
\begin{quote}
  \raggedleft
  Texto
\end{quote}
```

En español se refiere a la alineación como *bandera*, es decir, un texto alineado a la derecha tendrá *bandera a la derecha*, porque parece una bandera cuyo mástil está a la derecha.

8.2. Interlineado

Antes de describir cómo cambiar el interlineado en este extracto del artículo *Double-spaced documents in LaTeX* de la lista de preguntas frecuentes de LaTeX[1]:

> Of course, the real solution (other than for private copy editing) is not to use double-spacing at all. Universities, in particular, have no excuse for specifying double-spacing in submitted dissertations: LaTeX is a typesetting system, not a typewriter-substitute, and can (properly used) make single-spaced text even more easily readable than double-spaced typewritten text.

Según ese documento, lo mejor es usar el paquete `setspace` ya que mantiene el interlineado simple en los pies de figura y tabla o en las notas al pie, sitios donde no aporta nada que las líneas estén más separadas.

Para ello simplemente se carga el paquete y se elige el interlineado:

```
% Preámbulo
\usepackage{setspace}

\doublespacing % Interlineado doble
% \onehalfspacing % Interlineado 1.5
% \singlespacing % Interlineado simple
```

[1] Hay aplicaciones que permiten especificar un determinado formato como Writer2Latex

8.3. Márgenes

Para los márgenes, como en los demás casos, lo mejor es usar un paquete que los gestione. Uno apropiado es `geometry`.

Es fácil de usar: se carga el paquete especificando como argumento opcional el tamaño del margen. Para conseguir un margen uniforme:

```
\usepackage[margin=1cm]{geometry}
```

Y para definir cada margen por su lado:

```
\usepackage[top=1cm, bottom=1cm, right=0.5cm, left=1.5cm]{↩
    geometry}
```

Para más detalles siempre está disponible el manual

8.4. Sangría

Las normas tipográficas especifican que para separar dos párrafos se puede usar una línea en blanco o sangría, pero no ambas cosas. Si no se especifica nada, LaTeX usa la segunda opción. Para separar los párrafos con líneas en blanco lo más fácil es usar el paquete `parskip`. También se puede poner `\noindent` delante de cada párrafo que no queremos que se indente o usar `\setlength{\parindent}{0cm}`, pero el paquete `parskip` afecta al documento completo y evita problemas con la gestión del espacio.

Se carga en el preámbulo:

```
\usepackage{parskip}
```

8.5. Resumen

Como recomendación general a la hora de tocar el formato:

> Lo mejor es dejarle siempre a LaTeX las decisiones de estilo.
> Él sabe de tipografía.

Como recapitulación sobre la página:

- **Alineación**: LaTeX justifica por defecto pero se puede cambiar el alineado con entornos y comandos, `flushleft` y `\raggedright` alinean a la izquierda; `flushright` y `\raggedleft` a la derecha y `center` y `\centering` al centro.
- **Interlineado**: de por sí es simple y el paquete `setspace` ayuda a cambiarlo mediante órdenes como `\doublespacing`.
- **Márgenes**: se pueden modificar con el paquete `geometry` indicándole los valores de los márgenes al paquete como argumento opcional.
- **Sangría**: LaTeX sangra la primera línea de cada párrafo (excepto el primero) pero se puede modificar este comportamiento con el paquete `parskip`.

8.6. Referencias

LaTeX/Page Layout en WikiBooks

Quick note on line spacing

Paragraph indent and break en WikiBooks

`\raggedleft` en Hypertext Help with LaTeX

Capítulo 9

Espacio en blanco

LaTeX gestiona el espacio en blanco de forma automática. Esto tiene varias implicaciones:

- Da igual poner un espacio o muchos entre dos palabras, para Latex será un único espacio.
- Partirá las líneas donde mejor le venga a no ser que le obliguemos a hacerlo en un sitio determinado. Lo mismo puede decirse de las páginas.
- Dos trozos de texto que no estén separados por una línea en blanco pertenecerán al mismo párrafo a no ser que se especifique que no es así.
- No pintará una línea en blanco a no ser que le indiquemos expresamente que lo haga por muchas líneas en blanco que tengamos en el editor.

En cualquier caso, veremos cómo configurar aspectos del documento en contra de la lógica interna de LaTeX.

9.1. Espacio horizontal

Hay varias maneras de generar espacio en blanco en LaTeX. Algunas se describen a continuación, para más detalle hay más información en las referencias.

Primero los dos comandos que se describieron en el capítulo de ecuaciones:

- `\,` crea un espacio estrecho (`\thinspace`). Se usa cuando se desea separar ligeramente dos glifos porque interfieren entre sí.
- `~` crea un *espacio duro*, es decir, un espacio que no se puede partir. Es especialmente útil para las iniciales o casos como *Ejemplo 1* donde no se quiere que el uno vaya a otra línea[1].

[1] Por ese motivo el comando `\refeq` se ha definido usando un *espacio duro*

Hay más comandos de este tipo, cada uno con sus reglas de uso. Por ejemplo la pareja `\quad` y `\qquad` que equivalen respectivamente a 1 em[2] y a 2 em respectivamente.

Por otra parte hay un comando que permite introducir un espacio blanco del tamaño que se quiera:

```
\hspace{LONGITUD}
```

Esta longitud puede estar definida en cualquiera de las medidas que LaTeX entiende: puntos (*pt*), pulgadas (*in*), centímetros (*cm*), milímetros (*mm*), *em*, *ex*[3] o picas (*pc*).

Una cosa a tener en cuenta es que LaTeX ignorará el `\hspace{}` si viene justo después de un salto de línea ya que para él las líneas no pueden empezar con espacio en blanco. Ocurre exactamente lo mismo al final de las líneas. Se puede obligar a que ponga siempre el espacio con la versión con asterisco `\hspace*{}`:

```
% Diferencia entre \hspace y \hspace*
\hspace{1em}Línea alineada

\hspace*{1em}Línea movida 1 em hacia dentro
```

Un comando similar a `\hspace{}` es `\phantom{}`, que genera un espacio en blanco igual de ancho que su argumento:

```
% Hueco del tamaño de abc
\phantom{abc}
```

Finalmente tenemos `\hfill`, el *espacio de goma*. Este comando crea todo el espacio que puede dentro de una línea y sirve para situar cosas espaciadas pero con un cierto orden en la línea:

```
Texto a la izquierda \hfill Texto a la derecha
```

Todos estos comandos de espacio vienen bien a la hora de diseñar una portada para el documento o para crear encabezados y pies de página personalizados.

9.2. Espacio vertical

El espacio vertical se gestiona de manera muy similar al horizontal con los comandos `\vspace{}` y `\vfill`. El primero permite crear espacio vertical del tamaño deseado. Tiene una versión con asterisco para que LaTeX no ignore el espacio vertical tras un salto de página, es decir, al inicio de la página.

El segundo, `\vfill`, crea *espacio de goma* vertical y es especialmente útil cuando se quiere alinear texto verticalmente:

```
\vfill
Texto centrado verticalmente
\vfill
```

[2] Un espacio de la anchura de una letra *m* en la fuente actual
[3] Un espacio de la altura de una letra *x* en la fuente actual

9.3. Párrafos y saltos de línea

Para LaTeX un párrafo no acaba hasta que haya una línea en blanco o alguna indicación extra. Esta indicación es el comando `\par`, que apenas se usa a la hora de escribir ya que empeora la legibilidad y se suele reservar para definir entornos.

Hay que tener en cuenta que solo si se está usando el paquete `parskip` escribirá una línea entre los párrafos, sino, por mucho que en el editor los párrafos estén separados por una o varias líneas en blanco se verá dos párrafos juntos con el segundo de ellos sangrado.

Para provocar saltos de línea hay (entre otros) dos comandos `\\` y `\newline`, ambos comienzan una nueva línea pero no un nuevo párrafo. El primero es preferible dejarlo para alinear, ya que LaTeX lo redefine según el entorno, por ejemplo en las tablas, para organizar ecuaciones con `align` o tras un `centering`. El segundo solo puede usarse en el texto normal sin abusar en su uso.

9.4. Saltos de página

Por último, en cuanto los saltos de página, en general no hay que preocuparse de ello porque LaTeX tiene la habilidad de situar las cosas más o menos correctamente en la página. El uso de saltos de página es necesario en tres contextos:

1. Se quiere una página en blanco al inicio del documento para poner una dedicatoria, un resumen, los agradecimientos o algo similar.
2. Ha empezado una sección muy abajo en la página y no gusta cómo queda.
3. Ha puesto todas las imágenes en una página, la opción H de las figuras no funciona correctamente. Meter unos saltos de página adecuadamente puede solucionar ese tema.

Veamos las opciones que tenemos:

- `\newpage` provoca un salto de página dejando los párrafos que quedan tal cual, es decir, con espacio en la parte inferior como si se acabase el capítulo.
- `\pagebreak` provoca un salto de página y esparce los párrafos en la página para no dejar hueco blanco.
- `\clearpage` y `\cleardoublepage` provocan un salto de página y hacen que se pinten todos las figuras y tablas definidas hasta ese punto. La diferencia entre ellos es que, como su nombre indica, `\clearpage` salta una página y `\cleardoublepage` salta dos caras. Si estamos en un documento con diferentes estilos para la página derecha y la izquierda `\cleardoublepage` saltará las páginas necesarias para que la siguiente sea impar.

Una vez identificados los casos posibles y las diferentes opciones se puede combinar las dos cosas para obtener unas *reglas*:

- Para organizar secciones o figuras podemos elegir entre `\newpage`, `\pagebreak` y `\clearpage` dependiendo del resultado que queramos conseguir, si es a la mitad de un capítulo igual conviene más `\pagebreak`, cuestión a probar.
- Para las dedicatorias, resúmenes y similares de los libros la mejor opción es `\cleardoublepage` ya que lo va a poner siempre a la derecha como es lo habitual.

9.5. Resumen

Como resumen de este capítulo, incluyendo *buenas prácticas*:

- **Espacio horizontal**: poner un espacio en blanco o varios entre dos palabras produce el mismo resultado. Hay muchos comandos para crear espacio horizontal de mayor o menor tamaño, algunos para el texto, otros para las ecuaciones y algunos de ellos para ambas cosas. Si queremos crear un espacio a nuestro gusto tenemos `\hspace{}` y `\hspace*{}`, comandos que generan un espacio horizontal del tamaño que se especifique, siendo el del asterisco imposible de ignorar. Si se quiere un espacio que se ajuste solo tenemos `\hfill` *que es de goma*.
- **Espacio vertical**: LaTeX no va a crear espacio blanco vertical a menos que se lo digamos. Para ello tenemos los comandos `\vspace{}` y su versión con asterisco que fabrican espacio vertical del tamaño que le pidamos. Para situar elementos espaciados con cierto equilibrio en la página hay el *espacio de goma* `\vfill`.
- **Párrafos y saltos de línea**: lo más práctico es separar los párrafos con una línea en blanco y solo usar los saltos de línea en casos especiales, para ello tenemos `\newline` para usar en el texto normal y `\\` en los entornos especiales.
- **Saltos de página**: para saltar de página tenemos `\newpage`, `\pagebreak`, `\clearpage` y `\cleardoublepage`. Los tres primeros saltan a la siguiente página pero difieren en como tratan el texto y los objetos flotantes como las tablas y las figuras. El último salta dos caras y es útil para definir la página de dedicatoria o similares de un libro.

9.6. Referencias

Line breaks and blank spaces

What commands are there for horizontal spacing? en TexExchange

Whitespace character en la Wikipedia

When to use par *and when \, or blank lines* en TexExchange

LaTeX Line and Page Breaking

\pagebreak vs \newpage en TexExchange

Capítulo 10

Un documento científico

En este apartado se describe un **documento científico**, es decir libros técnicos, tesis, artículos....

La característica esencial de un documento científico es su *formato rígido* que en muchas ocasiones viene impuesto, ya sea por una revista o universidad o por las propias costumbres de la disciplina que tratamos. Por ejemplo, una tesis suele estar dividida en capítulos, debe tener referencias bibliográficas con un estilo determinado y se inicia con un índice general, uno de tablas y otro de figuras así como con un glosario de términos. También suele llevar un resumen del contenido al principio y se separa el trabajo adicional en apéndices. Todo esto implica diferentes formatos para las partes (numeración de las páginas y secciones, estilo de los encabezados...) y cierta planificación, así como el control de versiones.

Por ello primero veremos cómo organizar el documento y luego haremos un índice, un glosario y hasta unas lista de referencias.

10.1. División del documento

Lo primero que se hace es elegir el tipo de documento, empezando por preguntarnos si es un documento corto o largo ya que el formato del documento varía en gran medida según su longitud:

- **Documentos cortos**: para ellos se usa la clase `article` y sus derivadas, como `scrartcl` de las clases KOMA. Como norma general este tipo de documentos tienen 5 niveles de títulos[1] (`section`, `subsection`, `subsubsection`, `paragraph`, `subparagraph`), con los tres primeros numerados, y no llevan portada.
- **Documentos largos**: en este caso es conviniente usar las clases `book`, `report`[2] y sus derivadas como `memoir`. Estos suelen tener

[1] El uso de los diferentes niveles de título se vió en el primer documento.

[2] Estas dos clases sirven para objetivos diferentes: `report` está pensado para documentos no muy largos (trabajo de asignatura o parecido) y `book` para libros verdaderos. Hay varias diferencias entre ellas, por ejemplo, la clase `book` le da diferentes estilos a las páginas pares e impares por defecto mientras que `report` solo lo hace si lo especificamos; `book` abre los

7 niveles de títulos (`part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph`, `subparagraph`) de los que se numeran los cuatro primeros. Los demás niveles tienen un estilo pero ni se numeran ni aparecen en el índice. A diferencia de los documentos cortos, estos llevan portada por defecto.

Para el caso de un documento largo, si estamos usando las clases `book`, `memoir` o derivadas[3], aparte de los diferentes niveles de títulos tenemos unos comandos para identificar las distintas partes del documento y así darles estilos específicos. Todos ellos afectan desde que los escribimos hasta que aparece el siguiente:

- `\frontmatter` identifica las páginas preliminares. En esta parte los capítulos no llevan número y las páginas se numeran en romano. Es donde suelen ir el índice o los agradecimientos.
- `\mainmatter` da inicio al cuerpo del documento. Se numeran los capítulos y las páginas con números arábigos.
- `\appendix` como su nombre indica, especifica dónde empiezan los apéndices. Aquí los capítulos se *numeran* con letras pero se mantiene numeración de las páginas.
- `\backmatter` identifica las páginas finales. En esta parte los capítulos no se numeran y, como en los apéndices, se mantiene la numeración de las páginas. Sirve, por ejemplo, para las referencias bibliográficas.

Todo lo descrito sobre los estilos se puede modificar con relativa facilidad, pero sin tocar absolutamente nada se puede conseguir un documento con una estructura lógica aceptable.

Otro tema a tener en cuenta es la **división del contenido en diferentes archivos**. Cuando se escribe un documento largo este sistema no es muy apropiado. LaTeX permite escribir en diferentes archivos que luego se juntan para crear el documento final. De esta manera se puede tener un archivo con la definición del documento y el preámbulo desde el que se llama a otros archivos con el contenido propiamente dicho.

Para ello hay dos opciones:

- `\input{RUTA}` equivale a insertar el contenido del archivo ahí mismo. Tiene la ventajas de que se puede usar en cualquier parte (incluído el preámbulo) y que se puede anidar, es decir, el archivo *B* que se llama desde *A* puede a su vez llamar al archivo *C*.
- `\include{RUTA}` es un poco más complejo, que permite *compilar el documento a trozos* porque mantiene los números de página, secciones y demás como si se compilase el documento completo aunque solo se compile un trozo[4]. Más adelante se describen los archivos auxiliares. Es especialmente apropiado para capítulos ya que provoca un salto de página antes y después de incluir el contenido.

capítulos siempre a la derecha aunque pare ello tenga que dejar una página en blanco, `report` simplemente salta de página. Una comparación mas detallada en este artículo.

[3] `report` carece de estos comandos.

[4] Esto ocurre porque escribe archivos auxiliares para cada archivo que se llaman y luego lee a partir de ahí las referencias, números de página y otras cosas que varían

Sus desventajas son que no se puede anidar y que no se debe usar en el preámbulo.

A continuación se presenta un ejemplo de uso para entenderlo mejor:

```
\documentclass[a4paper,11pt]{book}

  % Cargamos el preámbulo que tenemos escrito en otro archivo
  \input{preámbulo}

  % Elegimos qué capítulos queremos que se compilen
  \includeonly{cap1,cap3} % sin espacios!

\begin{document}

  % Cargamos la portada desde el directorio Contenido
  \input{Contenido/portada}

  % Cargamos los capítulos desde el directorio Contenido.
  % Los capítulos capX.tex no llevan ni preámbulo ni definición del
  % documento, solo órdenes que irían tras \begin{document}

  \include{Contenido/cap1} % cargamos cap1.tex
  \include{Contenido/cap2} % cap2 no aparecerá en el resultado
  \include{Contenido/cap3} % mantiene la numeración como si cap2.tex estuviera incluido

\end{document}
```

Ya que tenemos comandos que permiten incorporar contenido desde otros archivos podemos aprovechar para organizar nuestro material en directorios para el contenido, el estilo o las imágenes. Incluso cabe la posibilidad de escribir las tablas largas en un archivo aparte y llamarlas con `\input{}` cuando corresponda.

10.2. Índices

Los índices de contenido, tablas, figuras o código se generan automáticamente. Solo es necesario escribir el comando correspondiente. La única cosa a tener en cuenta es que los índices no se incluyen por defecto en el índice de contenido, hay que *exigirle* a LaTeX que los añada.

Un ejemplo de cómo se crean los índices y cómo se incluyen en el índice es:

```
% Índice de contenido
\addcontentsline{toc}{chapter}{Índice general}
\tableofcontents

% Índice de tablas
\addcontentsline{toc}{chapter}{Índice de tablas}
\listoftables
```

```
% Índice de figuras
\addcontentsline{toc}{chapter}{Índice de figuras}
\listoffigures
```

Hay que tener en cuenta que hay que compilar el documento dos veces para que LaTeX fabrique los índices, la primera de ellas escribe los archivos auxiliares necesarios y la segunda los lee y crea el índice en concordancia.

Como todo en LaTeX, los índices se pueden personalizar. Por ejemplo, para el caso del índice de contenido, se puede cambiar la profundidad de los tres niveles que tiene por defecto al número que nos parezca mejor:

```
\setcounter{tocdepth}{NIVEL}
% NIVEL -1: part, 0: chapter, 1: section, etc.
```

También es interesante darle un título corto para los índices como argumento opcional a `\caption`, `\chapter` y otros comandos, por ejemplo:

```
\begin{figure}
  \caption[Título corto para el índice]{Título ultralargo ↩
      que describe la figura en el documento}
    \includegraphics{RUTA}
\end{figure}
```

Para acabar con los índices, si se usa el paquete `hyperref` los elementos de los índices se vuelven *clicables*. Hasta se puede poner los enlaces de colores:

```
\usepackage[colorlinks=true,linkcolor=magenta]{hyperref}
```

En el manual de `hyperref` especifica que es mejor que sea el último paquete que se carga por si acaso colisiona con algún otro.

10.3. Glosario y unidades

Otro tema que interesa en documentos científicos es poder hacer una lista de símbolos, letras griegas o acrónimos con facilidad.

Para el tema de los glosarios, nomenclaturas o listas de símbolos[5] es muy importante la planificación. *Muy importante*. Lo mejor es configurar todo antes de empezar a escribir, así se aprovecha las ventajas que tiene usar un paquete de este estilo. Se menciona *un paquete* porque hay unos cuantos para este propósito, aunque todos ellos tienen dos características en común:

- **Declaramos los símbolos al principio**. De este modo se garantiza la coherencia. Por ejemplo, si se ha llamado I a la corriente en el capítulo 3 y J en el 4. Además, al cambiar la declaración se modifican los símbolos correspondientes en todo el documento.

[5] Se usa *lista de símbolos* y *glosario* indistintamente para referirse a *una lista de cosas con su respectiva definición que se pone al inicio del documento para ayudar al lector a entender lo que se ha escrito*

- **La lista de símbolos se crea automáticamente**. Dependiendo del paquete tenemos la opción de crear listas de símbolos, letras griegas y acrónimos separadas o no, pero en todos los casos la lista se genera automáticamente de manera similar al índice de contenido y solo incluye los símbolos que se ha usado en el documento. Esto último permite tener declarados un muchos símbolos en un archivo aparte y tener la certeza de que solo se incluirán los que aparecen en el documento en el que estamos trabajando.

Veamos qué opciones hay. Por una parte están los que usan el programa `makeindex` lo que implica un paso extra a la hora de compilar: `nomencl` y `glossaries`, heredero del antiguo `glossary` y que tiene un manual de más de 200 páginas. Por otro, está `listofsymbols` que tiene menores capacidades pero no dificulta la compilación.

Aún con `listofsymbols` se puede añadir funcionalidades. Como ejemplo, de uso de `listofsymbols`:

```
\documentclass{article}

  \usepackage[draft]{listofsymbols} % cambiar 'draft' por '↩
      final' en el definitivo
  \begin{document}

    % Definición de símbolos [Definición]{Nombre que ↩
        usaremos}[Símbolo]
    \opensymdef
    \newsym[Energía]{symE}{E}
    \newsym[Masa]{symm}{m}
    \newsym[Velocidad de la luz]{symc}{c}
    % Hay que usar \ensuremath en los símbolos matemáticos
    \newsym[Longitud de onda]{symlam}{\ensuremath{\lambda}}
    \closesymdef

    % Crear lista de símbolos
    \listofsymbols

    % Uso en ecuaciones
    \begin{equation}
      \symE=\symm \symc^2
    \end{equation}

    % Uso en texto
    donde \symE es la energía \ldots

\end{document}
```

En lo que respecta a `nomencl` y a `glossaries`, en las referencias hay más información sobre ellos.

Algo parecido pasa con las unidades. En este caso se une al tema de la coherencia la tipografía, una de las mayores preocupaciones de los usuarios de LaTeX del mundo[6]

[6]Hay hasta una norma ISO sobre cómo dar formato a las constantes, variables, unidades y demás.

Si el principal problema es que las unidades queden bien, hay el paquete `units` que se ocupa de que las unidades sean tipográficamente correctas. Gracias a él las unidades tendrán el mismo estilo que el resto del texto en el modo texto y serán rectas en el modo matemático; no habrá nunca un salto de línea antes de la unidad, y habrá solo un *espacio estrecho* (`\,`) entre el valor y la unidad. Incluso crea fracciones gracias al paquete `nicefrac`. Cuenta con manuales muy detallados.

Simplemente se carga en el preámbulo:

```
\usepackage{units} % Opción 'ugly' para fracciones de texto ↩
    normal
```

Y se usan los comandos `\unit[VALOR]{UNIDAD}` y `\unitfrac[VALOR]{NUM}{DEN}` tanto en el modo texto como dentro de ecuaciones:

```
\unit[3]{m}

$\unitfrac[4]{m}{s}$
```

Si aparte de unidades bonitas se desea garantizar la coherencia en todo el texto tenemos el mucho más potente paquete `SIunits`. El enfoque de `SIunits` para las unidades es similar al de los paquetes para hacer glosarios y al que tiene el propio LaTeX para las ecuaciones: usar comandos para las unidades en lugar de texto normal. Es un paquete complejo, pero básicamente nos ofrece comandos para dar formato a números, unidades y números con unidades:

```
% Formato de número
\num[OPCIONES]{NÚMERO}

% Formato de unidad
\si[OPCIONES]{UNIDAD}

% Formato de valor con unidad
\SI[OPCIONES]{VALOR}{UNIDAD}
```

Se puede escribir las unidades por nosotros mismos o utilizar el comando correspondiente:

```
\si{kg}

\si{\kilogram}
```

Es interesante porque permite representar las fracciones como exponentes negativos y los prefijos como *kilo* o *mili* como potencias de 10:

```
\SI{10}{\kilo\metre\per\hour} % 10 km h^{-1}
\SI[per-mode = fraction]{10}{\kilo\metre\per\hour} % 10 km/h
\SI[prefixes-as-symbols=false]{10}{\kilo\metre\per\hour}  % ↩
    10 10^3 m h^{-1}
```

Todo esto se puede configurar en el preámbulo para no tener que escribir mucho. Muchas unidades tienen además nombres cortos para que no sean difíciles de definir.

10.4. Referencias bibliográficas

Para el manejo de las referencias bibliográficas se usa un programa diferente pero que va integrado en las distribuciones de LaTeX: el gestor de referencias bibliográficas BibTeX.

BibTeX permite tener una base de datos de documentos diversos que podemos citar en el texto y que luego listará donde se le indique con el estilo favorito. Lo mejor es que en esa lista solo aparecerán los elementos que se han citado y en el orden que se quiera independientemente del orden en el que se encuentren en la base de datos.

Además, tiene dos ventajas extra: todo va en texto plano y se pueden usar programas libres para generar la base de datos.

Hay otros paquetes más avanzados para gestionar la bibliografía como `natbib` y `biblatex` e incluso otros programas como Biber, especialmente desarrollado para trabajar con `biblatex`. En las referencias se listan otras herramientas.

Para gestionar la bibliografía es necesario lo siguiente:

- **Una bibliografía**, evidentemente, o bien la tenemos en un archivo `.bib` o la definimos en el propio documento.
- **Un estilo de cita** que vendrá definido en un archivo `.bst`, la propia distribución de LaTeX trae unos pocos pero se pueden descargar alguno que interese (por ejemplo, cuando una revista científica exige citar de una manera determinada) e incluso crear uno propio.
- Si vamos a tener la base de datos en un `.bib` vendrá bien un **programa de gestión bibliográfica**. Puede ser un programa específico como los libres Jabref o Zotero o un editor de uso general como Emacs.
- **Compilar 4 veces** en una secuencia `pdflatex`, `bibtex`, `pdflatex`↩, `pdflatex`[7]. Se considera `pdflatex` pero puede ser perfectamente `xelatex` o simplemente `latex`. Si se usa un IDE se hará automáticamente.

Para gestionar la base de datos bibliográfica lo más simple es usar un programa con su GUI y guardar todo un `.bib`. Este archivo no deja de ser texto plano, de hecho, si se abre con un editor de texto normal se observa que cada una de las entradas de la base de datos tiene un aspecto como este:

```
@Book{Perez2018,
  title={Curso de LaTeX},
  author={Perez, Adán},
  year={2018},
  note=\url{https://perezadan.com/cursoLatex/}
}
```

Siempre es posible definir la bibliografía directamente en LaTeX:

[7] El proceso que se sigue es algo así: el primer comando escribe las claves de las referencias en el archivo auxiliar. A continuación `bibtex` lee esas claves junto con el `.bib` y el `.bst` y crea la bibliografía en un archivo `.bbl`. El segundo `latex` escribe las referencias cruzadas en el `aux`. Por fin, el último escribe las referencias cruzadas en el archivo final.

```
\begin{thebibliography}{9} % Menos de 9 entradas

\bibitem{Perez2018}
  Adán Pérez,
  \emph{Curso de LaTeX},
  2018.

\end{thebibliography}
```

En estos dos ejemplos `Perez2018` es la *clave* de la entrada bibliográfica y es lo que se usa para citar ese trabajo en el documento mediante el comando `\cite{CLAVE}`:

```
Tal y como dice \cite{Perez2018}
```

Si se ha escrito la bibliografía a mano con `thebibliography`, en el documento sustituirá `\cite{Perez2018}` por un número. Si se usa un archivo `.bib`, en cambio, tenemos ventajas añadidas ya que podemos pedir un estilo de cita y de referencia bibliográfica concreto. Este estilo está definido en un archivo `.bst`. En el documento se listan los que BibTeX tiene por defecto.

En definitiva, es necesario dos líneas:

```
% Definimos el estilo bibliográfico
\bibliographystyle{plain}

% Creamos bibliografía a partir de referencias.bib
\bibliography{referencias}
```

En las referencias hay más información sobre la bibliografía.

10.5. Resumen

La planificación es fundamental a la hora de escribir un documento científico. La planificación se extiende tanto a la organización de los ficheros como a la propia escritura del documento, ya que elegir una manera de hacer las cosas antes de hacerlas ayuda luego a la hora de trabajar. Por eso es conveniente hacerse las siguientes preguntas antes de escribir el documento:

- ¿Qué tipo de documento nos conviene? ¿Hay alguna clase que facilite la labor?
- ¿Necesitamos crear una lista de símbolos? ¿Vamos a hacerla a mano o es preferible usar un paquete? En este último caso, ¿usamos uno sencillo o incluimos una etapa extra de compilación para ganar en funcionalidad?
- ¿Podría ser interesante un paquete para tratar las unidades? ¿Queremos solo que sean bonitas o también cierta coherencia?
- Si nuestro documento tiene referencias bibliográficas ¿cómo vamos a crear la base de datos? ¿Qué programa vamos a utilizar?

También una buena idea para escribir un documento largo es dividirlo en un archivo en el que se define su esqueleto y otros en los que está el contenido propiamente dicho. El esqueleto tendrá un aspecto similar a este:

```
% Definición del documento
\documentclass[opciones]{tipo}

% Paquetes
\usepackage[opc]{paquete}

% Datos
\title{título}
\author{autor}
\includeonly{cap1}

% Inicio
\begin{document}
  % Título e índices
  \maketitle

  \frontmatter
  \tableofcontents
  \listoftables
  \listoffigures

  % Contenido
  \mainmatter
  \include{cap1}
  \include{cap2}

  % Apéndices
  \appendix
  \include{ap1}

  % Referencias bibliográficas
  \backmatter
  \bibliographystyle{plain}
  \bibliography{referencias}
  % Fin
\end{document}
```

10.6. Referencias

\documentclass{book, report, article or letter} en texblog

Regarding the book, report, and article document classes: what are the main differences? en TexExchange

LaTeX Notes: Structuring Large Documents

What is the right order when using \frontmatter, \tableofcontents, \mainmatter, \part, \chapter, \backmatter,\appendix etc? en TexExchange

When should I use \input vs. \include? en TexExchange

10 ways to customize toc/lof/lot

The glossaries package v4.25: a guide for beginners (pdf)

LaTeX glossary and list of acronyms

Nomenclatures en ShareLaTeX

LaTeX tips: Bibliographies

Bibliography in LaTeX with Bibtex/Biblatex

Tame the BeaST (pdf)

Bibliography management with `natbib` en ShareLaTeX

`biblatex` *in a nutshell (for beginners)* en TeXExchange

A BibTeX Guide via Examples (pdf)

Debibify, un colección de estilos de cita

Capítulo 11

Insertando código

En este capítulo se describe el *resaltado de sintaxis*, es decir, a darle formato al código fuente que se haya insertado en el documento con la idea de que sea más fácil de leer.

Hay varios paquetes que permiten escribir con colores el código: `listings`, `minted` y `LGrind`. En este apartado se describen los dos primeros.

11.1. Lo fácil: listings

El paquete `listings` se utiliza de manera similar al resto de paquetes que se ha visto hasta ahora: se carga, se especifica sus opciones y luego se utilizan los comandos que proporciona en el cuerpo del documento.

Para incluir el código hay varias opciones. Por una parte, igual que ocurría con las ecuaciones y las figuras, se puede escribir código en la propia línea (*inline*) o puede flotar. Y también como en el caso de las figuras y ecuaciones, al código flotante se le puede poner un *pie de código* (`caption`) y etiquetarlo (`label`) para luego referenciarlo en el texto (`\ref{}`).

Para incluir código *inline* tenemos el comando `\lstinline`, muy útil él para hacer referencia a órdenes y variables en el texto:

```
Guardamos el resultado en la variable \lstinline!fib!
```

Los signos de exclamación delimitan el código y pueden sustituirse por cualquier otro carácter no incluido en el código.

Para escribir código con línea propia está el entorno `lstlisting`, al que le podemos dar como opciones tanto el *pie de código* y su posición como la etiqueta del trozo de código:

```
% Código con pie en la parte inferior (por defecto en la ↩
    superior) y etiqueta
\begin{lstlisting}[language=haskell, caption=Código con ↩
    Listings, captionpos=b, label=lst:fiboHaskell]
  -- Fibonacci!
  fib = 0 : 1 : zipWith (+) fib (tail fib)
\_end{lstlisting}
```

Por otra parte, se puede escribir el código en el propio documento o importarlo desde un archivo externo.

```
\lstinputlisting[language={[latex]tex}]{Codigo/codigo.tex}
```

Este ejemplo de LaTeX tiene una característica especial: es un *dialecto* de TeX, para ver cuál es la sintaxis para indicar dialectos.

Este comando es especialmente interesante con los argumentos opcionales `firstline` y `lastline` mediante los cuales le indicamos a LaTeX el trozo de programa que nos interesa pintar:

```
\lstinputlisting[language={[latex]tex}, firstline=5, ↩
    lastline=10]{Codigo/codigo.tex}
```

Al igual que con el entorno `lstlisting`, al código importado mediante `\lstinputlsiting` se le puede poner un *pie de código* con el argumento opcional `caption`. Por cierto, si queremos, se puede recopilar todos los listados de código flotantes en un índice con el comando `\lstlistoflistings`, del mismo modo que se hace con las tablas y las figuras.

Para personalizarlo. Si no se especifica nada, `listings` escribirá los comentarios en cursiva, las palabras claves en negrita y demás, pero todo ello en blanco y negro. Si queremos otro tipo de formato o usar diferentes colores necesitamos indicárselo según lo indicado a continuación.

Primero se carga el paquete `xcolor` que, como se ha visto antes permite usar colores en el documento:

```
\usepackage[usenames,dvipsnames,svgnames,table]{xcolor}
```

Después, en el cuerpo del documento se define el estilo del código con `\lstset{OPCIONES}`. Un ejemplo con diferentes opciones (hay muchas, lo mejor es mirar en el manual) es:

```
\lstset{
    tabsize=2, % tab = 2 espacios
    backgroundcolor=\color[HTML]{F0F0F0}, % color de fondo
    captionpos=b, % posición de pie de código, b=debajo
    basicstyle=\ttfamily, % estilo de letra general
    columns=fixed, % columnas alineadas
    extendedchars=true, % ASCII extendido
    breaklines=true, % partir líneas
    prebreak = \raisebox{0ex}[0ex][0ex]{\ensuremath{\↩
        hookleftarrow}}, % marcar final de línea con flecha
    showtabs=false, % no marcar tabulación
    showspaces=false, % no marcar espacios
    keywordstyle=\bfseries\color[HTML]{007020}, % estilo de ↩
        palabras clave
    commentstyle=\itshape\color[HTML]{60A0B0}, % estilo de ↩
        comentarios
    stringstyle=\color[HTML]{4070A0}, % estilo de strings
}
```

Se usa `\bfseries` en lugar de `\textbf` porque *ya estamos dentro de un grupo*.

También se tiene la opción de definir diferentes estilos mediante el comando `\lstdefinestyle{NOMBRE}{ESTILO}` para luego aplicarlos con la opción `style`. Por ejemplo para resaltar mejor el código de Python que de un estilo general que se aplican a todos los listados de código y luego unas pocas opciones más que solo se aplican cuando son necesarias:

```
\lstdefinestyle{abaqusPython}{
    language=python,
    % Palabras clave extra
    morekeywords={CONTINUOUS,NUMBER,MESH,par,name,ParStudy,
    template,define,sample,combine, generate},
    % Delimitadores extra, s porque hay uno a cada lado
    moredelim=[s][\ttfamily\color{magenta}]{<}{>},
}

\lstinputlisting[style=abaqusPython]{calculo.py}
```

Por último, hay que tener en cuenta que `listings` no acepta UTF8 por lo que no va a escribir los caracteres acentuados directamente. Para solucionar esto lo más sencillo es crear un `\lstset` con la opción `literate`, que sirve para sustituir elementos en el código. En sí, `literate` se usa para que el código pueda leerse con mayor facilidad, pero viene bien para que cada vez que vea `á` lo sustituya por `\'a` y queden bien los acentos. Una versión con más características se encuentra aquí:

```
% Listings no acepta UTF8
{original}{sustitución}{tamaño}
\lstset{literate=
  {á}{{\'a}}1
  {é}{{\'e}}1
  {í}{{\'i}}1
  {ó}{{\'o}}1
  {ú}{{\'u}}1
  {Á}{{\'A}}1
  {É}{{\'E}}1
  {Í}{{\'I}}1
  {Ó}{{\'O}}1
  {Ú}{{\'U}}1
  {ñ}{{\~n}}1
  {ü}{{\"u}}1
  {Ü}{{\"U}}1
}
```

La principal debilidad del paquete `listings` es la de no ser un *lexer* completo, es decir, que no distingue todos los elementos del lenguaje. Para solucionar este problema tenemos el paquete `minted`.

11.2. Lo no tan fácil: minted

El paquete `minted` utiliza Pygments, un resaltador de sintaxis general escrito en Python, para dar formato al código. Está basado además en el paquete `fancyvrb` (*Fancy Verbatim*), un paquete para dar formato al texto *verbatim*, es decir, al *texto tal cual, sin interpretar las marcas*.

No es tan fácil porque requiere una etapa de compilación intermedia y requiere tener Python 2.6 o superior y Pygments instalados. Ejemplo:

- Hay disponibles un mayor número de lenguajes. ¡Hay hasta resaltador de BrainFuck!
- Es más fácil de personalizar
- Acepta UTF8, al menos con `xelatex` y `polyglossia`, y hay posibilidad de elegir la codificación a partir de su versión 2.0
- Está bien mantenido y hay mejoras desde 2016.

Evidentemente, también tiene sus desventajas:

- Hay que compilar con la opción `-shell-escape` para que LaTeX permita la ejecución de un programa externo, lo que implica ciertos problemas de seguridad.
- La versión[1] de los repositorios suele ser antigua, con lo que perdemos las últimas funcionalidades como la de elegir la codificación. Además, actualizarlo requiere `fvextra` y en algunos repositorios no se encuentra.

Para usarlo la opción más básica es usar el entorno `minted` junto con el lenguaje del código en cuestión:

```
\begin{minted}{python}
 for n in range(10):
   if n%2:
     print n
\end{minted}
```

Que tiene la versión reducida `\mint` si el código que queremos añadir tiene solo una línea. Para delimitar el código se puede usar diferentes caracteres, los elegimos según los delimitadores que use nuestro código.

```
\mint{python}|print [n for n in range(10) if n%2]|
```

Tanto para el caso del entorno `minted` como para el comando `\mint` el código tendrá línea propia, pero no se le puede considerar flotante, para ello debemos rodear cualquiera de las dos opciones con el entorno `listing`, más abajo hay un ejemplo completo en el que se muestra cómo hacerlo.

Este paquete también permite incluir código *inline*:

```
Guardamos el resultado en la variable \mintinline{python}{↩
    fib}
```

Y también tenemos la opción de importar el código tal y como hacíamos con `listings`:

```
\inputminted{python}{fibonacci.py}
```

Lo más difícil de usar de `minted` es llamar al comando que compilará en documento con la opción `-shell-escape`, por ejemplo:

```
pdflatex -shell-escape documento.tex
```

[1] Podemos ver la versión del paquete que estamos usando en el archivo *.log*

Para personalizar el código hay diferentes opciones:

- Con `\usemintedstyle[LENGUAJE]{ESTILO}` elegimos el estilo de Pygments con el queremos dar formato al código. Como veis el lenguaje es opcional, podemos elegir un estilo para cada lenguaje si se desea. Para ver los estilos instalados escribimos `pygmentize -L styles` en el terminal. También podemos definir nuestro propio estilo siguiendo las directrices de Pygments.
- Los comandos `\setminted[LENGUAJE]{ESTILO}` y `\setmintedinline[LENGUAJE]{ESTILO}`, que tiene precedencia, también valen para definir los estilos. El primero afecta al entorno `minted` y al comando `\mint` y el segundo al código *inline*.
- El comando `\newminted[NOMBRE]{LENGUAJE}{OPCIONES}` nos permite guardar las opciones para un determinado lenguaje para el entorno `minted`. Es similar a `\lstdefinestyle` del paquete `listings`. Si no le ponemos nombre llamará `LENGcode` a nuestro nuevo estilo donde `LENG` es el lenguaje del mismo. Para usarlo no tenemos más que sustituir el nombre del lenguaje por el del estilo que hemos definido en el entorno `minted`.
- El comando `\newmint[NOMBRE]{LENGUAJE}{OPCIONES}` es el equivalente a `\newminted` para el comando `\mint`. Funciona exactamente igual excepto por que sustituye `\mint` por el nombre que le hemos dado al estilo.

```
\newmint[py]{python}{bgcolor=black}

\py|print [n for n in range(10) if n%2]|
```

En cuanto a las opciones muchas coinciden con las de `listings`, lo mejor es ir al manual y mirar.

Un ejemplo completo tiene las siguientes características:

- Es un objeto flotante con opción de posición H (*here*)
- Crea un índice de codigo con `\listoflistings`, el equivalente a `\lstlistoflistings`
- Activa los números de línea con la opción `linenos`
- Nos deja escribir ecuaciones en los comentarios del código gracias a `mathescape`
- Nos permite escribir en LaTeX dentro de los comentarios del código gracias a `texcl`. Esta opción se ha convertido en `texcomments` a partir de la versión 2.0. En este caso se usa para poder enlazar la página web.

```
% Cambiamos Listing por Listado
\renewcommand{\listingscaption}{Listado}
\listoflistings

\begin{listing}[H]
  \begin{minted}[linenos,mathescape,texcl]{clojure}
  ;; Fibonacci por cortesía de \href{https://pfctelepathy.wordpress.com/}{Fulano}
```

```
  ;; $F_n = F_{n-1} + F_{n-2} \,/\, F_0 = 0 \wedge F_1 = 1$
  (defn fibo
      ([] (fibo 0 1))
      ([one two]
          (lazy-seq (cons one (fibo two (+ one two))))))
  \end{minted}
  \label{lst:fibo}
  \caption{Código con Minted}
\end{listing}
```

Que quedaría así:

```
1     ;; Fibonacci por cortesía de Ekaitz
2     ;; F_n = F_{n-1} + F_{n-2} / F_0 = 0 ∧ F_1 = 1
3
4     (defn fibo
5         ([] (fibo 0 1))
6         ([one two]
7             (lazy-seq (cons one (fibo two (+ one two))))))
```

Listado 1: Código con Minted

Figura 11.1: Ejemplo de uso de minted

Como nota final, `listings` no sabe resaltar Clojure y obligaría a crear nuestro propio *resaltador*.

11.3. Resumen

En este capítulo se ha visto cómo resaltar código de dos maneras, una de ellas usa un paquete de la forma tradicional y la otra llama a un programa externo. Ambas dan una funcionalidad similar siendo tal vez `listings` más fácil de usar y `minted` más fácil de personalizar. Los dos paquetes tienen muchas opciones que podemos consultar en sus respectivos manuales.

11.4. Referencias

Manual de `listings`

`minted` en GitHub

Manual de `minted`

Code listings in LaTeX en el FAQ de LaTeX

Code listing en ShareLaTeX

`minted` vs `textments` vs `verbments` en TeXExchange

Are Text-Only Data Formats Safe? Or, Use This LaTeX Class File to Pwn Your Computer

Capítulo 12

Presentaciones

Con LaTeX además de hacer documentos con una excelente calidad también se puede crear presentaciones. Para ello hay varias clases diferentes, `beamer` es la más famosa, pero también tienen el mismo objetivo `powerdot` y las más antiguas `prosper`, `seminar` y `slides`. Aquí se describe la clase `beamer`.

12.1. ¿Merece la pena usar LaTeX para una presentación?

Ventajas:

- **Contenido y formato separados**: esta es una de las características fundamentales de LaTeX y aquí resulta especialmente útil, definimos ambas cosas por separado y se afectan muy poco entre sí.
- **Orden lógico**: es obligado escribir el contenido como si fuera un texto y no como unos cuadrados con cosas dentro.
- **Formato favorable para el espectador**: es más complicado poner muchísimo texto o imágenes sin necesidad en una diapositiva que hacerla sencilla y clara.
- **Texto plano**: como siempre, trabajamos con texto plano por lo que no es necesario un programa específico[1], el resultado no depende del sistema operativo, la colaboración más sencilla y demás ventajas habituales del texto plano que ya conocemos.
- **Reutilización**: si la presentación deriva de otro documento, como un artículo o tesis, que está escrito en LaTeX se puede copiar el trozo correspondiente a las imágenes, ecuaciones, tablas... directamente en la presentación.

También tiene inconvenientes:

[1] A la hora de presentar tal vez sea necesario un programa si queremos usar alguna funcionalidad específica.

- **No vemos lo que hacemos**: esto puede ser un problema para una presentación ya que es algo más visual. Este problema es especialmente acuciante si no tenemos claro el orden en el que queremos decir las cosas.
- **Diseños complejos**: es difícil crear una diapositiva con muchos elementos y que siga teniendo buena apariencia. Esto quiere decir que nunca conseguiríamos reproducir presentaciones comerciales que tienen en cada página el logo de la empresa y su lema, un índice de contenidos, varias imágenes y tablas y texto en tres tipos de fuente diferente. LaTeX es más minimalista.

12.2. La clase beamer

Describamos `beamer`. A pesar que esta clase tenga un manual de cientos de páginas enseguida uno puede montarse una presentación decente. Esto se debe a que las tablas, imágenes, bibliografía, apéndices[2], ecuaciones y demás características de LaTeX funcionan exactamente igual, aunque su aspecto varía dependiendo del estilo que estemos utilizando. Del mismo modo, `\maketitle` produce la portada y `\tableofcontents` genera el índice de contenidos como viene siendo habitual.

En definitiva, se crea el documento de la misma manera que creamos un artículo o libro, definiendo el contenido de cada diapositiva dentro del entorno `frame`:

```
\begin{frame}{Título}
  % Contenido de la diapositiva
\end{frame}
```

Algo a tener en cuenta es que tanto las secciones como las subsecciones añaden una entrada al índice pero no establecen el título de la diapositiva, se debe hacer manualmente. Dependiendo del estilo, tener diferentes secciones y subsecciones permite crear diapositivas de título para separar cada sección y que la presentación pueda seguirse más fácilmente.

12.2.1. Estilo

En cuanto al estilo se debe diferenciar dos cosas: los **temas** y el **estilo de ciertos elementos** como las alertas, los ejemplos o los teoremas, que cada tema redefine.

El **tema** es lo que establece el formato general de la presentación, vendría a ser como la plantilla. Aparte de los temas que define el propio `beamer` y de los que veremos a continuación, se tienen muchos temas disponibles en Overleaf. Algunos ejemplos de temas interesantes son: *Presento*, el de la universidad de Berkeley y Metropolis.

Lo más importante respecto a los temas `beamer` es que hay cinco tipos:

- **Temas de presentación**: afectan a toda la presentación. Eligen un tema de color, uno de fuente, uno exterior y otro interior que

[2] Para tener apéndices numerados es necesario cargar el paquete `appendixnumberbeamer`

combinen (relativamente) bien. Tienen nombres de ciudades. Antes se llamaban cosas tipo `bars` o `shadow`.

- **Temas de color**: afectan a la paleta de colores de la presentación. Hay temas de color *exterior*, con nombres de animales acuáticos; *interior*, con nombres de flores; y completos con nombre de animales voladores. Los temas interiores afectan al color del interior de la diapositiva; los exteriores a los elementos del borde y los completos a ambas cosas.
- **Temas de fuente**: controlan el tipo de fuentes que se usan en la presentación, por defecto es Sans Serif, pero hay opción de Serif (`serif`); títulos en negrita (`structurebold`), títulos en cursiva (`structureitalicserif`) y títulos en versalita (`structuresmallcapsserif`).
- **Temas interiores**: controlan el aspecto de los elementos del interior de la diapositiva, es decir, cómo se muestran los bloques, las tablas, las figuras, las listas... Las opciones son `default`, `circles`, `rectangles`, `rounded` e `inmargin`. Lo más fácil es probar que hace cada una.
- **Temas exteriores**: controlan el aspecto de los bordes, es decir, el encabezamiento, el pie y la barra lateral. Las opciones son `default`, `miniframe`, `sidebar`, `split`, `shadow`, `tree` y `smoothtree`. También se puede probar las diferentes opciones.

Se pueden combinar como parezca para que la estética sea mejor. Existen sitios que matrices recopilan combinaciones de estilos, sobre todo para los temas de presentación y de color.

Los temas se establecen así:

```
% Preámbulo
\usetheme{Bergen} % tema de presentación
\usecolortheme{rose} % tema de color
\usefonttheme{serif} % tema de fuente
\useinnertheme{circles} % tema interior
\useoutertheme{split} % tema exterior
```

En caso de que haya la posibilidad de definir más opciones, se añaden entre corchetes como argumento opcional, dependiendo de cada tema.

En cuanto al **estilo de los diferentes elementos** de la presentación, son los temas los que establecen cómo son todos y cada uno de ellos, por lo que si no es conforme, por ejemplo, la apariencia que tienen las listas se puede sobreponer su estilo por defecto el propio. Esto permite que la presentación sea coherente. En el siguiente apartado se describe cómo se modifica el estilo de los elementos.

12.2.2. Opciones

Lo que sí cambia respecto a otros documentos de LaTeX es el modo de establecer las opciones, ya que para ello usamos la familia de comandos `\setbeamer` y especificamos qué elemento queremos cambiar y cómo. Según lo que queramos conseguir tenemos diferentes comandos:

- `\setbeameroption{opción general}`, establece las opciones generales para la presentación. Por ejemplo y tal y como veremos en la siguiente sección, se usa para especificar a `beamer` que muestre u oculte las notas mediante `\setbeameroption{hide notes}`.
- `\setbeamertemplate{elemento}{definición}`, define el aspecto de cierto elemento, por ejemplo, con `\setbeamertemplate{itemize item}{$\Rightarrow$}` se consigue que en las listas no numeradas se indiquen los ítems con una viñeta de flecha.
- `\setbeamercolor{elemento}{fg=colorPrimerPlano, bg=colorDeFondo}`, establece el color de determinado elemento, por ejemplo, `\setbeamercolor{title}{fg=magenta, bg=white}` establece que todos los títulos sean rosas con el fondo blanco. No es necesario usar las opciones `fg` y `bg` a la vez, lo que no se cambia mantendrá el color que tenía.
- `\setbeamerfont{elemento}{size=tamaño, shape=estilo}`, establece la forma y tamaño de fuente de determinado elemento, por ejemplo, `\setbeamerfont{title}{series=\bfseries}` pone en negrita el título de la presentación.Se puede establecer solo el tamaño o solo el estilo, lo que no cambiemos permanecerá como estaba.

Para ver qué elementos tiene una presentación no queda otra que acudir al manual, pero ahora ya tenemos los elementos para entenderlo perfectamente. También en el manual encontraremos los argumentos opcionales de estos comandos.

Podemos usar todos estos comandos en cualquier parte del documento y afectan desde donde están situados hasta encontrarse con otra definición o, si no hay ninguna más, hasta el final.

12.2.3. Notas

Con `beamer` se tiene la opción de crear unas notas secretas que solo ve el presentador en la línea de la consola del presentador de Impress. Para escribir las notas se usa el comando `\note{}` que crea una página de notas detrás de la diapositiva en cuestión:

```
\documentclass[notes=show]{beamer}
\begin{document}
  \begin{frame}
    % Contenido de la diapositiva
    \note{Notas}
  \end{frame}
\end{document}
```

Esto es interesante, pero se le puede sacar mucho más provecho uniéndolo al paquete `pgfpages` que permite unir la diapositiva con la página de notas en una hoja más ancha de tal manera que al proyectarla el orador vea las notas y la audiencia la presentación. Hay que tener en cuenta que esta funcionalidad no es compatible con todos los visores de *pdf*.

El comportamiento de las notas se controlan mediante las siguiente opciones:

- `\setbeameroption{hide notes}` solo muestra la presentación.

- `\setbeameroption{show only notes}` solo muestras las notas.
- `\setbeameroption{show notes on second screen=right}` crea una presentación el doble de ancha que contiene las diapositivas y las notas. En este caso se muestran las notas en la pantalla de la derecha, con `left` las pondríamos en la izquierda.

Recopilando queda:

```
\documentclass{beamer}

\usepackage{pgfpages}
\setbeameroption{show notes on second screen=right}

\begin{document}
  \begin{frame}
    % Contenido de la diapositiva
    \note{Notas}
  \end{frame}
\end{document}
```

12.2.4. Efectos y multimedia

Una presentación en `beamer` no significa que sea aburrida y estática, se puede personalizar cómo va apareciendo el contenido e incluso añadir multimedia.

Overlay Mediante las opciones de *overlay* se puede controlar cuando se muestra cada elemento. Esto viene bien, por ejemplo, para mostrar los elementos de una lista uno a uno. Para ello LaTeX creará múltiples copias de la diapositiva con *overlays* que mostrarán los elementos que vayamos indicando. De esta manera, al ir avanzando dará la sensación de que va *surgiendo* o *desapareciendo* contenido en la presentación.

Hay varios comandos para gestionar este mecanismo, como los que siguen:

- `\pause`: solo se muestra el contenido hasta este punto.
- `\uncover<OVERLAY>{CONTENIDO}`: solo se muestra el contenido en las copias indicadas pero se le reserva espacio desde el principio.
- `\only<OVERLAY>{CONTENIDO}`: como `\uncover` pero sin que se reserve espacio previamente para el contenido.

Además, muchos otros comandos y entornos aceptan opciones de *overlay*, generalmente con esta estructura:

```
\comando<OVERLAY>{CONTENIDO}

\begin{entorno}<OVERLAY>
CONTENIDO
\end{entorno}
```

En `OVERLAY` especificamos cuándo queremos que aparezcan las cosas, `<1>` mostrará el contenido solo en la primera copia; `<2->` en todas a partir de la segunda y `<-5>` solo hasta la quinta.

Un caso interesante es el de las listas, ya que hay una sintaxis simplificada para mostrar los elementos de uno en uno:

```
\begin{itemize}[<+->]
 \item Ítem 1
 \item Ítem 2
\end{itemize}
```

Los *overlays* también funcionan en las notas:

```
\note<1>{Notas para el ítem 1}
\note<2>{Notas para el ítem 2}
```

Vídeos En las presentaciones de `beamer` también puede contener vídeos. Hay varios paquetes para este fin, como `multimedia`, que viene con el propio `beamer` y `media9`, que sustituye a `media15`. El problema es que pocos visores de *pdf* soportan los vídeos incrustados. Si encontramos uno, los vídeos se incrustan muy fácilmente:

```
% Vídeo con multimedia
\movie[OPCIONES]{SUSTITUTO}{VÍDEO}

% Vídeo con media9
\includemedia[OPCIONES]{SUSTITUTO}{VÍDEO}
```

Donde `SUSTITUTO` es el texto o imagen que guardará sitio al vídeo, por ejemplo, un fotograma del mismo. En `VÍDEO` se debe escribir la ruta al vídeo.

Otra opción es usar el comando `\href` del paquete `hyperref`, que sirve para crear enlaces en nuestros documentos. En lugar de incrustar el vídeo en la presentación, en este caso ponemos un texto o imagen para que cuando la pinchemos se abra el reproductor de vídeo en una ventana aparte:

```
\href{run:VIDEO}{SUSTITUTO}
```

Navegación En la parte inferior de las diapositivas aparecen por defecto unos iconos para navegar por la presentación y buscar. Si no se desean se eliminan con:

```
\setbeamertemplate{navigation symbols}{}
```

Repetición Aquí se describe cómo insertar automáticamente un contenido concreto cuando se dé cierto evento en `beamer`. Esto es útil para hacer aparecer una diapositiva con el título o por poner un par de ejemplo, para mostrar el índice cada vez que vaya a comenzar una nueva sección.

Esto se consigue con la familia de comandos`\AtBegin` (y `\AtEnd` para el caso de las notas) que se activan al comenzar una sección, parte, nota o demás. Por ejemplo:

```
% Diapositiva con el título de sección al iniciar sección
\AtBeginSection{
  \begin{frame}
  \vfill
  % Caja con colores de título
  \begin{beamercolorbox}[center]{title}
    \usebeamerfont{title} % Fuente de título
    \insertsectionhead % Nombre de sección
  \end{beamercolorbox}
  \vfill
  \end{frame}
}

% Índice mostrando subsección actual al iniciar subsección
\AtBeginSubsection
{   \begin{frame}{Outline}
        \tableofcontents[currentsection,
        currentsubsection,
        sectionstyle=show/hide,
        subsectionstyle=show/shaded/hide]
    \end{frame}
}
```

12.3. Programas para presentar

El mayor problema de `beamer` es que no todos los visores de *pdf* son capaces de mostrar las notas y proyectar las diapositivas. Si se usan las presentaciones como están, con cualquier lector en pantalla completa sirve. No obstante hay diferentes alternativas para que poder leer las notas secretas. En concreto *pdfpc*.

12.3.1. Pdfpc

Pdfpc es una herramienta de línea de comandos para visualizar presentaciones en formato *pdf* en varias pantallas. Es un *fork* de *Pdf Presenter Console*, que dejó de desarrollarse. Se distribuye con licencia GNU GPL v2 así que es software libre. Es fácil de utilizar y ayuda mucho a la hora de presentar, no solo por las notas.

La única desventaja es que hay que compilarlo desde el código fuente porque el de los repositorios es muy antiguo, no es difícil, se puede hacer en GNU/Linux y hasta en Windows con Cygwin.

Para usarlo simplemente escribimos:

```
pdfpc PRESENTACIÓN
```

Donde `PRESENTACIÓN` es la ruta a la presentación en *pdf*.

De por sí *pdfpc* nos enseña en la vista de presentador la diapositiva actual, la siguiente, un reloj, el número de la diapositiva actual y el total. Todo ello muy útil a la hora de presentar.

Para ver las notas de `beamer` es necesario crear la presentación con las notas integradas como hemos visto antes:

Figura 12.1: pdfpc en acción

```
\setbeameroption{show notes on second screen=right}
```

Luego llamamos a *pdfpc* con la opción `--notes`:

```
pdfpc presentation.pdf --notes=right
```

Ahora en la vista de presentador veremos las notas y una minidiapositiva mostrando la diapositiva actual, en lugar de verla en grande como antes.

El programa tiene otras muchas opciones, un resumen de algunas se describen, las demás están en el manual:

- `-d`, `--duration=N` la duración en minutos (`N`) de la presentación. Sirve para que ponga una cuenta atrás en la parte inferior de la pantalla.
- `-l`, `-- lastminutes =N` tiempo en minutos (`N`) a partir del que la cuenta atrás se verá en rojo.
- `-s`, `-- switchscreens` cambia la vista de presentador de pantalla.
- `-w`, `--windowed` crea dos ventanas, una con la vista del presentador y otra con lo que verá la audiencia. Útil para ver el resultado cuando solo tenemos una pantalla.

Por ejemplo, para presentación de una tesis:

```
pdfpc presentation.pdf --duration=45 --notes=right --last-↩
    minutes=10
```

Además, durante la presentación se pueden usar diferentes teclas para *hacer cosas*:

- `F` (*freeze*): congela la imagen de la presentación para la audiencia mientras el orador navega en su vista. Coloca un copo de nieve en la parte inferior.

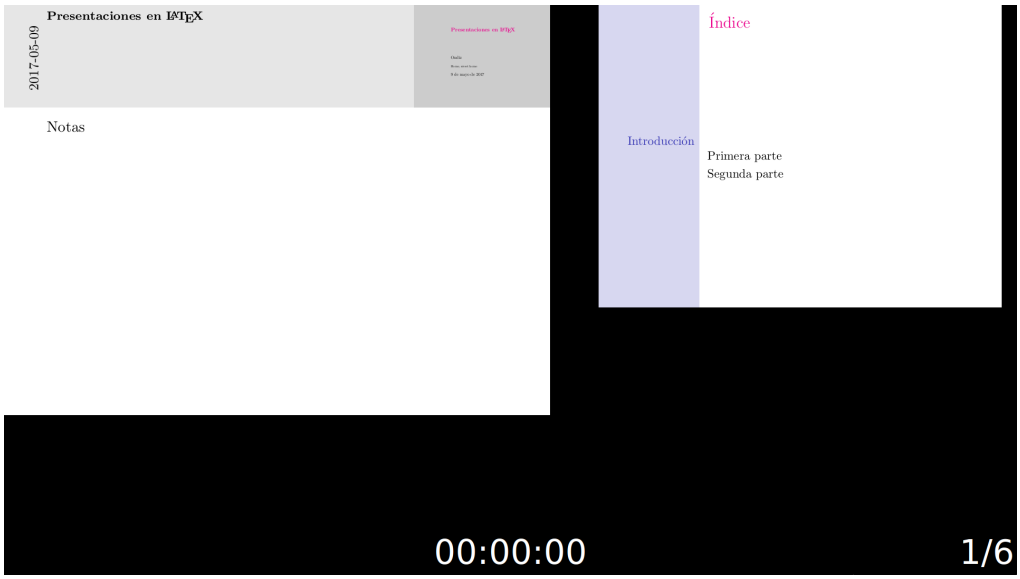


Figura 12.2: pdfpc con notas

- `B` (*black*): pone la pantalla de la audiencia negra y al orador muestra un cuadradito negro con una cruz blanca. Útil cuando se da clase y se alterna pizarra y proyector.
- `G` (*go*): nos lleva a la diapositiva que le indiquemos.
- `N` (*notes*): nos permite escribir notas en la diapositiva. Salimos con `ESC`.
- `E` (*end*): marca la diapositiva final. Útil si tenemos *diapositivas de repuesto* para las preguntas.
- `O` (*overlay*): sirve para marcar/desmarcar diapositivas como parte de una diapositiva que va surgiendo poco a poco. No las tendrá en cuenta en el cómputo total de diapositivas.
- `P` (*pause*): pausa el reloj.
- `R` (*reset*): reinicia la presentación.
- `Q` (*quit*) o `ESC`: cierra la presentación.

Las notas y diferentes marcas (fin, *overlay*, ...) las guarda en un archivo *pdfpc* que recupera cada vez que leemos la presentación. Es un archivo de texto plano y se puede abrir. Tiene esta apariencia:

```
[file]
presentation.pdf
[duration]
45
[skip]
8,
[end_user_slide]
10
[notes]
### 1
```

```
Notas en la diapositiva 1
```

12.3.2. Otras opciones para presentar

Otros programas son:

- *Impressive*: un programa escrito en Python con funcionalidades interesantes como el modo foco y la vista global de todas las diapositivas.
- *Pympress*: también escrito en Python, soporta vídeo y las notas de `beamer` y tiene una consola para el presentador.
- *Dspdfviewer*: un visor simple para las presentaciones de `beamer`.

12.4. Resumen

Para resumir veamos cómo quedaría un ejemplo completo aunque sencillo de una presentación en LaTeX:

```
% Definición
\documentclass{beamer}

% Notas
\usepackage{pgfpages}
\setbeameroption{show notes on second screen=right}

% Datos
\title{Presentaciones en \LaTeX}
\author{Pedro}
\institute{Home, sweet home}
\date{\today}

% Temas
\usetheme{Bergen}
\usefonttheme{serif}
\usecolortheme{rose}

% Opciones
\setbeamercolor{title}{fg=magenta, bg=white}
\setbeamertemplate{navigation symbols}{}

% Inicio
\begin{document}

% Diapositivas
 \begin{frame}
  \maketitle
  \note{Notas}
 \end{frame}

 \begin{frame}{Índice}
  \tableofcontents
  \note{Más notas}
 \end{frame}
```

```
\section{Introducción}
\subsection{Primera parte}

\begin{frame}{Introducción}
 \begin{itemize}
  \item<1-> Ítem 1
  \item<2> Ítem 2
 \end{itemize}
\end{frame}

\end{document}
```

En cualquier caso, para empezar con `beamer` se recomienda coger una presentación ya hecha y probar a cambiar cosas hasta que estemos cómodos con este nuevo sistema de trabajo.

Una presentación de ejemplo se encuentra en el directorio de Ejemplos.

12.5. Referencias

Which package to use for presentations? Beamer, Prosper, or Other en TeXExchange

Creating a presentation with LaTeX and powerdot

Manual de `beamer`

Producing beautiful slides with LaTeX

Beamer theme matrix

Beamer theme gallery

Create your own beamer template

Embedding videos and animations en TeXExchange

Is there a nice solution to get a "presenter mode" for Latex presentations? en TeXExchange

Capítulo 13

Macros

Este capítulo trata sobre la creación de *macros*, es decir, de nuevos comandos y entornos[1]. Hay un par de ideas que hay que tener claras a la hora de definir cosas en LaTeX.

Lo primero y más importante para crear macros en LaTeX es que hay dos opciones [2]:

- **Crear un entorno o comando desde cero**. Con ello se consigue que LaTeX haga algo que no hacía o se guardan un conjunto de órdenes que se usan a menudo en una macro con el objetivo escribir menos. La palabra clave para esto es *new*.
- **Sobreescribir un entorno o comando existente**. En este caso la idea es modificar el comportamiento de cierto comando o entorno a nuestro gusto. Se conoce como *renew*.

El siguiente concepto en orden de importancia es que podemos (re)definir comandos en cualquier parte del documento, pero para mejorar la organización es preferible hacerlo en el preámbulo.

A continuación se muestra cómo crear comandos y entornos nuevos y modificar los existentes.

13.1. Escribir comandos

Han ido apareciendo comandos nuevos[3] y trucados[4] anteriormente.

13.1.1. Comandos nuevos

Definir comandos nuevos en LaTeX es sencillo, solo se debe seguir la siguiente estructura:

[1] LaTeX es un *conjunto de macros* para TeX. Se ha separado las *macros* en comandos y entornos, pero un entorno no deja de ser un conjunto de comandos que afecta de forma local. Llamaremos también *macro* a los comandos (y por extensión entornos) propios que *se definan*

[2] también está `\providecommand`, que crea el comando si no existe y sino ignora la definición.

[3] En el capítulo de ecuaciones hay una macro para no tener que escribir la palabra *Ecuación* a la hora de referenciar.

[4] En el capítulo de idioma hay una macro para modificar el nombre de las tablas.

```
\newcommand{COMANDO}[ARGUMENTOS]{DEFINICIÓN}
```

donde:

- `COMANDO` será el nombre del comando que queremos definir. Empezará por `\`. Solo podemos usar letras para bautizarlo.
- `ARGUMENTOS` será el número de argumentos entre 0 y 9 que le pasaremos al comando. Que un comando tenga argumentos es opcional.
- `DEFINICIÓN` será donde escribimos lo que hace el comando. Haremos referencia a los diferentes argumentos mediante `#` seguida del número correspondiente.

Por ejemplo para crear un comando que escriba `Figura X` en lugar de `X` cuando se haga referencia a cierta figura:

```
\newcommand{\figref}[1]{\figurename~\ref{#1}}
```

Analizándolo:

- El nombre del nuevo comando es `\figref{}` y tiene un único argumento, la etiqueta de la figura, a la que hacemos referencia con `\ref{}`.
- `\figurename` es el comando que guarda el nombre de las figuras[5]. Podríamos escribir *Figura* a mano, pero si cambiamos el idioma tendríamos que cambiar también la definición. De este modo LaTeX, sustituye `\figurename` por el nombre de la figura según el paquete de idioma.
- Usamos un *espacio duro* entre el nombre y el número para que LaTeX no inserte en medio un salto de línea o de página.

Esta nueva macro se usa exactamente igual que cualquier otro comando:

```
\begin{figure}[H]
  \includegraphics[width=0.7\textwidth]{Figuras/esquema.eps}
  \caption{Esquema del proceso}
  \label{fig:esquema}
\end{figure}

Como vemos en la \figref{fig:esquema}...

% Equivalente a
Como vemos en la Figura~\ref{fig:esquema}...
```

Un tema interesante a la hora de definir comandos es la inclusión de **argumentos por defecto** en la definición del mismo:

```
\newcommand{COMANDO}[ARGUMENTOS][DEFECTO]{DEFINICIÓN}
```

donde `DEFECTO` es el valor que tomará el argumento opcional si no se especifica. El argumento opcional siempre es el primero.

Un ejemplo de uso podría ser un texto matemático en el que se hace referencia al plano real a menudo pero que haga falta alguna ocasión

[5] Del mismo modo, el nombre de los capítulos se guarda en `\chaptername` y el de las tablas en `\tablename`. No todos los elementos siguen este patrón, de hecho, `\sectionname` no existe.

hacer referencia de un espacio de dimensión mayor. Para ello se puede definir un comando que escriba la R del conjunto de números reales y que por defecto el espacio sea bidimensional, pero podamos cambiarlo opcionalmente:

```
\usepackage{amssymb}
\newcommand{\R}[1][2]{\mathbb{R}^{#1}}
```

A la hora de usarlo se utiliza el argumento opcional cuando sea necesario:

```
% Espacio bidimensional
$\R$

% Espacio tridimensional
$\R[3]$
```

13.1.2. Comandos trucados

Además de crear nuestros propios comandos podemos modificar el comportamiento de alguno existente. Para ello usamos `\renewcommand` en lugar de `\newcommand` con la misma sintaxis:

```
\renewcommand{COMANDO}[ARGUMENTOS]{DEFINICIÓN}
```

donde:

- `COMANDO` será el nombre del comando que queremos modificar.
- `ARGUMENTOS` será el número de argumentos, igual que antes.
- `DEFINICIÓN` será la nueva definición del comando.

Como ejemplo vamos a usar `\renewcommand` para evitar tener que activar el modo matemático cuando escribamos fracciones. Con este fin usamos `\ensuremath{}`, que permite usar comandos matemáticos dentro y fuera de las ecuaciones.

Como se va a crear la nueva definición a partir del propio comando, necesitamos guardarlo en otro sitio primero para que la definición no sea recursiva. Para ello ayuda el comando `\let`, que sirve para copiar el contenido de un comando en uno nuevo:

```
\let\comandoNuevo=\comandoViejo
```

Juntando las piezas, se obtiene lo siguiente:

```
% Guardamos la definición original
\let\oldfrac=\frac
% Modificamos \frac para que funcione fuera de ecuaciones
\renewcommand{\frac}[2]{\ensuremath{\oldfrac{#1}{#2}}}
```

Ahora se puede usar `\frac` directamente en el texto.

13.2. Escribir entornos

En lo que sigue se muestra cómo crear y modificar entornos.

13.2.1. Entornos nuevos

La sintaxis para la definición de entornos nuevos es muy similar a la de los comandos, usando ahora `\newenvironment`:

```
\newenvironment{ENTORNO}[ARGUMENTOS]{ANTES}{DESPUÉS}
```

En `ANTES` escribiremos el grupo de comandos que hay que ejecutar al iniciar el entorno y, por tanto, los que le darán el formato al mismo, y `DESPUÉS`, los que se activarán tras el texto. El resto de elementos funciona como antes.

De esta manera, podemos definir un entorno para poner notas en el texto. Por ejemplo uno que rodea el texto de la nota con dos rayas (`\hrule`), una por debajo y una por encima, y que permite darle un título, que aparecerá en negrita:

```
\newenvironment{nota}[1]
  {\vspace{1ex}\hrule\textbf{#1}}
  {\vspace{1ex}\hrule}
```

Este nuevo entorno se usa en el cuerpo del documento como cualquier otro:

```
\begin{nota}{ Cuidado !}
  Hay que tener en cuenta que
\end{nota}
```

Un tema interesante son los contadores gracias a los cuales se pueden generar **entornos numerados**. Los contadores tiene la estructura `\theELEMENTO`. Así, `\thepage` contiene el número de página, `\thechapter`[6] el número de capítulo y `\theNOMBRE` el número del elemento numerado que hayamos creado.

Para crear un contador se usa el comando `\newcounter`, le damos un nombre y, opcionalmente, se indica dónde debe reiniciar la cuenta:

```
\newcounter{NOMBRE}[REINICIO]
```

El tema del reinicio es interesante para los documentos largos, así podemos empezar a contar al iniciar un capítulo y hacer referencia al elemento mediante `\thechapter.\theNOMBRE` que escribirá el número del capítulo seguido del número del elemento.

Luego, se incrementa el valor del contador cuando sea necesario con:

- `\stepcounter{CONTADOR}`: incrementa en uno el valor de `CONTADOR`.
- `\refstepcounter{CONTADOR}`: incrementa en uno el valor de `CONTADOR` y permite usarlo en las referencias cruzadas.
- `\addtocounter{CONTADOR}{NÚMERO}`: incrementa `CONTADOR` en un valor que le pasemos.

Una idea para hacer uso de esta funcionalidad es crear un entorno numerado para poner ejemplos en el texto cuya numeración se reinicie al cambiar de sección:

[6] Uniéndolo con lo descrito anteriormente, `\thechapter` contiene el número del capítulo, `\chaptername` su nombre y `\chaptermark` el título.

```
% Creamos un nuevo contador que se reinicie al cambiar de ↩
    sección
\newcounter{ejemplo}[section]
% Incrementamos en uno el contador al iniciar el nuevo ↩
    entorno
% Accedemos a su contenido con \theejemplo
\newenvironment{ejemplo}
  {\refstepcounter{ejemplo}\vspace{1ex}\hrule\textbf{Ejemplo↩
      ~\thesection.\theejemplo}}
  {\vspace{1ex}\hrule}
```

Así, si escribimos algo de este estilo:

```
\section{Entorno numerado}
  \begin{ejemplo}
    Un primer ejemplo
  \end{ejemplo}

  \begin{ejemplo}
    Un segundo ejemplo
  \end{ejemplo}
```

Conseguimos lo siguiente:

1. Entorno numerado

Ejemplo 1.1 Un primer ejemplo

Ejemplo 1.2 Un segundo ejemplo

Figura 13.1: Entornos numerados

13.2.2. Entornos trucados

Al igual que se modifican comandos existentes con `\renewcommand`, se pueden cambiar entornos con `\renewenvironment`:

```
\renewenvironment{ENTORNO}[ARGUMENTOS]{ANTES}{DESPUÉS}
```

Es importante tener en cuenta la implementación original del entorno que vamos a cambiar. Se puede pensar en un entorno como dos comandos, uno que da inicio al formato concreto y otro que lo finaliza. Es decir, esto:

```
\begin{equation}
  a^2 x + b x + c = 0
\end{equation}
```

es equivalente a:

```
\equation
  a^2 x + b x + c = 0
\endequation
```

De esta manera resulta más sencillo saber qué hay que escribir en los argumentos ANTES y DESPUÉS.

Un ejemplo complejo con entornos para que las citas aparezcan en gris oscuro con una barrita gris clara a la izquierda es:

```
\usepackage{framed}

% Redefinir leftbar
\renewenvironment{leftbar}[1][\hsize]
  {\color{gray}
    \def\FrameCommand
    {{\color{lightgray}\vrule width 3pt}}
    \MakeFramed{\hsize#1\advance\hsize-\width\FrameRestore}
  }
  {\endMakeFramed}

% Guardar entorno quote, lo forman dos comandos
\let\oldquote=\quote
\let\oldendquote=\endquote

% Barra vertical a la izquierda de la cita
\renewenvironment{quote}
  {\vspace{10pt}\leftbar\vspace*{-6pt}\oldquote}
  {\oldendquote\endleftbar\vspace{10pt}}
```

La única manera de aprender a modificar entornos es modificar entornos lo que significa practicar.

13.3. Una nota sobre TeX

LaTeX es un conjunto de macros escritos en TeX (o Plain TeX). Esto provoca que LaTeX tenga algunas limitaciones a la hora de definir cosas, limitaciones que TeX, al ser de más bajo nivel, no tiene.

El comando `\let` que hemos usado para guardar la definición original de un comando que íbamos a modificar pertenece a TeX. También `\def` que aparece en ejemplo de las citas personalizadas es parte de TeX y sirve para definir comandos nuevos, al igual que `\newcommand`[7].

13.4. Lo que hay que saber

Conocer la diferencia entre *definir* (`\new`) y *sobreescribir* (`\renew`) un comando o entorno existente y cuándo hay que hacer lo uno o lo otro. Tampoco está demás recordar que escribimos tanto las definiciones como las modificaciones en el preámbulo. Igualmente, viene bien saber de la existencia de comandos de TeX como `\let` y `\def`.

Un ejemplo final es el proceso para crear macros:

1. Escribir la combinación de comandos en el cuerpo del documento y verificar que funciona.

[7] Se pueden declarar comandos nuevos dentro de entornos de manera similar a declararlos de manera independiente. Aquí hay más información.

2. Escribir cómo se quiere que sea el comando o entorno final y compararlo con la combinación de comandos.
3. Trasladar la combinación al preámbulo y dar forma según la sintaxis correspondiente.
4. Extraer los argumentos y analizar si se puede dar un valor por defecto a alguno de ellos.
5. Probar si funciona.

13.5. Referencias

LaTeX example: How to create your own commands with `newcommand`

LaTeX/Macros en Wikibooks

newcommand with arguments in LaTeX

Plain TeX vs.~LaTeX Macros en TeXExchange

LaTeX/Plain TeX en Wikibooks

List of higher-level LaTeX commands corresponding to TeX commands en TeXExchange

What is the difference between
def and
newcommand? en TeXExchange

Capítulo 14

Caja de herramientas

Este capítulo presenta herramientas externas y trucos que pueden ayudar a la hora de crear un documento.

14.1. Fase de pruebas

Las partes del proceso de testar el formato: crear un borrador y generar un documento de prueba en el que podemos cambiar el formato sin necesidad de tener contenido.

14.1.1. Un borrador

Podemos crear un borrador del documento gracias al argumento opcional `draft` de `\documentclass`. Tiene dos ventajas:

- **Se reduce el tiempo de compilación** ya que no se pinta todo el contenido. Por ejemplo, se sustituyen las imágenes por cajas del mismo tamaño que contienen el nombre de la imagen.
- **Permite ver más fácilmente errores** como las *overfull boxes*[1], lugares en los que el texto invade el margen, que marca con una barra.

Muchos paquetes tienen esta opción, cada cual con sus características concretas. Si establecemos `draft` como opción de la clase afecta a todos ellos, si queremos que solo afecte a alguno, podemos darla como opción del paquete.

14.1.2. Texto e imágenes de prueba

Existen un par de paquetes que rellenan las páginas de texto sin sentido para que nos resulte más fácil ver el efecto de algún paquete u opción. El más sencillo es `lipsum`, la implementación para LaTeX del típico *Lorem Ipsum*. Es fácil de usar:

[1] Las *overfull boxes* ocurren a menudo porque LaTeX no sabe cómo partir una palabra (pasa mucho con las direcciones de correo electrónico y las URLs), en esos casos hay que hacerlo manualmente

```
\documentclass[draft]{article}
\usepackage{lipsum}
\begin{document}
  \lipsum % Texto Lorem Ipsum completo
  \lipsum[17] % El párrafo 17 será texto sin sentido
\end{document}
```

Otra opción un poco más potente es `blindtext`, que tiene varias ventajas respecto a `lipsum`:

- Depende del idioma así que se puede ver el proceso de silabación. Ahora mismo los idiomas disponibles son inglés, francés, alemán, latín y catalán.
- A diferencia del paquete `lipsum`, que solo crea texto, tenemos la posibilidad de generar listas, ecuaciones y hasta un documento completo si se lo pedimos.
- Podemos crear texto formado por pangramas, frases que contienen todas las letras del abecedario de un idioma concreto, interesante para probar tipografías. También podemos usar trozos de la Biblia.

Veamos ahora cómo se usa, que es un poco más complejo que `lipsum`. Para crear texto tenemos dos comandos:

- `\blindtext` crea un trozo de texto corto. Opcionalmente le podemos dar el número de veces que queremos que repita el texto con `\blindtext[REPETICIONES]`.
- `\Blindtext` crea un trozo de texto largo. Puede tomar como opciones el número de párrafos y el de repeticiones con `\Blindtext[PARRÁFOS][REPETICIONES]`

Podemos generar un documento completo, largo o corto, con y sin ecuaciones:

- `\blindocument` crea un documento corto, si delante activamos las matemáticas con `\blindmathtrue` tendrá también ecuaciones. El documento tendrá títulos de sección de todos los niveles y varios tipos de listas.
- `\Blindocument` crea un documento largo, funciona exactamente igual que el anterior.
- `\blindmathpaper` crea un documento con ecuaciones.

Hay otras opciones para crear listas y así, en el manual se encuentra detallado.

En cuanto a las **imágenes**, el paquete `mwe` (*Minimal Working Example*) proporciona diversas imágenes de prueba[2] de diferentes proporciones y formatos. Este paquete también carga `graphicx` y si están instalados, `lipsum` y `blindtext`. No hace falta que carguemos el paquete para usar estas imágenes, las instala de tal manera que al compilar se pueda acceder a ellas.

Para usarlas simplemente se llaman por su nombre:

[2] Se encuentra en `/usr/share/texlive/texmf-dist/tex/latex/mwe/`

```
\includegraphics{example-image-a}
```

14.2. Una plantilla

Cuando el formato está definido, y lo vamos a usar a menudo, podemos crear una plantilla a partir del documento. Todos los IDEs permiten exportar una plantilla a partir del documento actual, que más adelante podremos importarla y tendremos una base desde la que trabajar.

Otra opción es crear nuestra propio paquete (*sty*) o clase (*cls*). Crearemos un paquete si nuestras macros valen para diferentes clases, y si, en cambio,estamos definiendo un tipo de documento, crearemos una clase. Tanto las clases como los paquetes consisten en lo que escribimos en el preámbulo de nuestros documentos formateado de otra manera.

Un ejemplo de paquete que carga los paquetes de idioma para cuando queremos escribir en español consistiría en el archivo `español.sty` con el siguiente contenido:

```
% Versión de LaTeX usada
\NeedsTeXFormat{LaTeX2e}[1994/06/01]

% Nombre del paquete creado
\ProvidesPackage{español}

% Paquetes usados
\RequirePackage[utf8]{inputenc}
\RequirePackage[spanish]{babel}
\RequirePackage[T1]{fontenc}
\RequirePackage{lmodern}
```

En este caso usamos `\RequirePackage` en lugar de `\usepackage`, pero se parece bastante a un preámbulo habitual. Guardamos nuestro paquete en un lugar accesible y lo llamamos como a cualquier otro con `\usepackage{español}`.

Hay más información sobre la creación de clases y paquetes en las referencias.

14.3. Cambios y versiones

Que manejemos texto plano tiene una gran ventaja: podemos fácilmente incluir software de control de versiones en nuestro flujo de trabajo. Además de las ventajas que ya tiene el control de versiones de por sí, nos permite incluir información sobre las versiones y los cambios en el propio documento

Tenemos, además, muchos paquetes para añadir información sobre los cambios al documento. Algunos incluyen al pie o en una marca de agua la versión actual, como `gitinfo2`, con otros como `vhistory` registramos el historial de cambios en una tabla de manera manual mediante el comando `\vhEntry`:

```
\vhEntry{VERSIÓN}{FECHA}{AUTOR}{CAMBIOS}
```

Después, podemos hacer referencia a esta información en el cuerpo del documento, ya sea el número de versión o la lista de autores. Aquí está el manual del paquete.

Es fácil crear **registro de cambios** simples si usamos `git`. Una manera es etiquetar los puntos más interesantes, exportar una tabla con la información deseada e incluirla en el documento mediante el comando `\input{}`.

Para etiquetar podemos usar una etiqueta *anotada* o *ligera*, con la primera podemos añadirle un mensaje a la etiqueta, la ligera simplemente marca el *commit* anterior. Por ejemplo para usar etiquetas ligeras:

```
git tag NOMBRE
```

Luego, para exportar en formato LaTeX la información de cada etiqueta que hemos creado usamos algo de este estilo, dependiendo de qué queramos conseguir:

```
git for-each-ref --format="%(refname:short)␣&␣%(authordate:↩
    short)␣&␣%(subject)␣\\\\" refs/tags > log.tex
```

Como hemos formateado la información como si fuera el contenido de una tabla, lo incluimos dentro de un entorno `tabular` en el que escribimos los encabezados:

```
\begin{tabular}{l l l}
  \textbf{Versión} & \textbf{Fecha} & \textbf{Descripción} ↩
      \\
  & & \\
  \input{log.tex} % incluir datos
\end{tabular}
```

Por supuesto, podríamos crear una macro, formatear la información de cualquier otro modo o incluso fusionarlo con el paquete `vhistory`.

Para automatizar esto hay varias maneras, una de ellas es escribir un *hook* de `git`, otra es incluir la línea anterior en el Makefile, si se usa, y la más simple es modificar la orden de compilación que ejecuta el IDE.

Una manera interesante de **mostrar los cambios** entre dos versiones de un documento es `latexdiff`, un *script* de Perl que compara dos archivos de LaTeX y genera un tercer archivo destacando los cambios.

Una vez instalados Perl y `latexdiff`, no hay más que escribir lo siguiente en la terminal[3]:

```
latexdiff viejo.tex nuevo.tex > dif.tex
```

Si compilamos `dif.tex` conseguiremos algo como aparece en la Figura 14.1

Tacha en rojo lo que hemos borrado y destaca en azul lo que hemos añadido. Parece especialmente útil para cuando estamos colaborando, se ve de un vistazo cómo quedan los cambios.

[3] Si estáis en Windows y después de instalar Perl no funciona, se prueba `latexdiff-so`, la versión *standalone*.

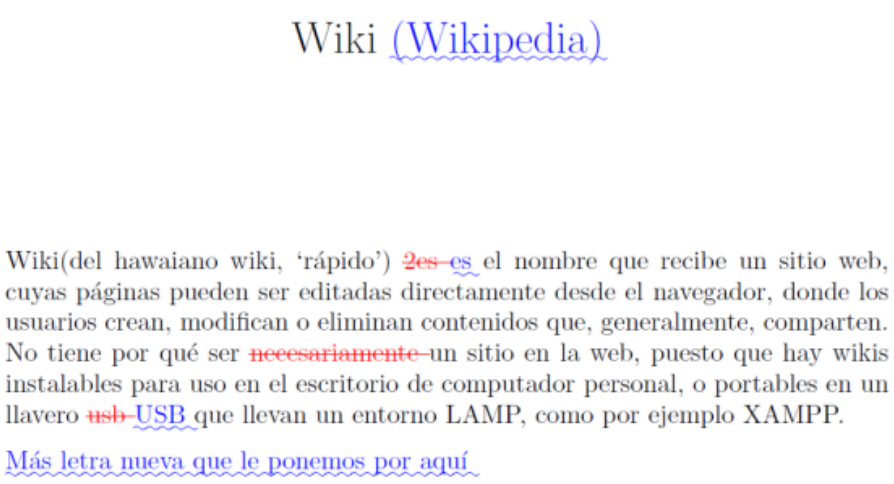


Figura 14.1: Marcando las diferencias

Si no se quiere instalar nada también hay una versión online en la que pegamos el contenido de los archivos a comparar y aparece el archivo de diferencias que podemos a su vez copiar y compilar.

14.4. Exportar a LaTeX

Este último apartado sobre las herramientas trata de exportar contenido que tenemos en otros formatos a LaTeX. De esta manera conseguimos dos cosas:

- Incluir con facilidad contenido creado en otros programas en LaTeX.
- Ayudarnos a crear elementos que resultan complejos de llevar a cabo en LaTeX, como podrían ser las tablas y las portadas.

En el caso de las tablas se entrelazan ambas problemáticas, cuesta escribir tablas en LaTeX y, además, muchas veces tenemos esas tablas en otros formatos. Si nuestro problema es el primero, tenemos *editores online* con los que crear la tabla de manera visual y exportarla a continuación. Si ocurre un poco de las dos cosas hay varias herramientas, como *Excel2Latex*, un *plugin* para Excel que permite seleccionar y exportar una tabla concreta en formato LaTeX. También hay una versión para OpenOffice, *Calc2LaTeX*, y tenemos *Matrix2LaTeX* para Python y Matlab.

Una herramienta similar es *Writer2LaTeX* que sirve para exportar de Libre Office Writer a LaTeX, no es la panacea pero puede ayudar cuando tenemos que seguir cierta plantilla y solo nos proporcionan un *doc*.

El programa definitivo para las labores de conversión es *Pandoc*. Con una terminal y una orden muy sencilla transforma casi cualquier formato de texto a casi cualquier otro:

```
pandoc OPCIONES INPUT -o OUTPUT
```

14.5. Resumen

En este capítulo se han descrito diferentes herramientas útiles sin entrar demasiado en detalle y analizar las que mejor se adapten a nuestras necesidades. Hemos descrito borradores, texto e imágenes de prueba, plantillas, clases y paquetes, control de versiones y sobre cómo exportar el contenido creado en otros programas a LaTeX. Evidentemente, hay muchas más herramientas relacionadas con LaTeX.

14.6. Referencias

What does the draft mode change? en TeXExchange

Creating a LaTeX Minimal Example (pdf)

El paquete `lipsum`

El paquete `blindtext`

Example images in LaTeX? en TexExchange

Creating a default preamble en TeXExchange

Manual de Kile (pdf)

Manual de TexStudio

LaTeX/Creating Packages en Wikibooks

LaTeX 2ε for class and package writers

Best practice for maintaining change history in tex en TeXExchange

Tracking changes in LaTeX files

Registro de cambios en un documento LaTeX con git

LaTeX packages for use with revision control

Highlighting the different between Latex Documents in Windows 7 using Latexdiff

LaTeX/Collaborative Writing of LaTeX Documents en Wikibooks

When and why should I use % !TEX TS-program and % !TEX encoding? en TeXExchange

Converting a spreadsheet to LaTeX

Apéndice A

Nota sobre los archivos auxiliares

Cuando compilamos LaTeX desde un IDE se generan varios archivos auxiliares al generar el documento.

Una lista de los archivos auxiliares típicos, todos ellos son texto plano así que se pueden abrir y ver qué contienen:

- `aux`: sirve para gestionar las referencias cruzadas (`\ref`) y las citas bibliográficas (`\cite`), entre otras cosas. En general, guarda la información que ha de pasarse de un proceso de compilación a otro.
- `log`: aquí se guarda la información sobre el proceso de compilación, las advertencias, los errores, los paquetes utilizados con su respectiva versión y tal. Es especialmente útil si falla al crear el documento y no se sabe por qué.
- `synctex`: sincroniza nuestro código fuente con el documento de salida en el IDE, es decir, al movernos por el código fuente vemos al lado la parte correspondiente del *pdf*. Si se borra simplemente se creará otro.
- `toc`, `lof`, `lot` ... : sirven para crear el índice, la lista de figuras, la lista de tablas ...
- `out`: lo genera `hyperref` y vale para crear los enlaces que nos llevan de un lado a otro en el *pdf*.
- `bbl`: es el archivo de bibliografía creado por BibTeX.
- *Otros*: otros paquetes crean archivos auxiliares para gestionar sus cosas, por ejemplo, el paquete `listofsymbols` crea un `sub` y un `sym` para gestionar la lista de subíndices y de símbolos respectivamente. Otro ejemplo son los archivos `idx` que sirven para hacer el índice por palabras con `makeindex`.

Para el caso de Beamer se crean además estos otros:

- `snm`: ayuda en el proceso de insertar imágenes en el tipo `article`

- `nav`: sirve para crear la barra de navegación en la presentación

Todos estos archivos aparecen porque cuando se crea el documento, por dentro en realidad se invocan varios procesos. Tomando como ejemplo un caso en el que usemos `pdflatex` y tengamos referencias se haría algo así:

```
pdflatex fuente.tex   # primera compilación
bibtex fuente.aux     # la bibliografía
pdflatex fuente.tex   # para referencias
pdflatex fuente.tex   # para referencias
```

Como se observa, es necesario compilar tres veces para tener las referencias correctamente. A este proceso se le pueden añadir otros pasos como `makeindex`, para el índice por palabras, o `makeglossaries`, para crear el glosario. Es por este conjunto de procesos que nosotros no vemos que se escriben todos los archivos auxiliares.

Si no se quiere tener esos ficheros auxiliares hay varias opciones:

- Si se usa un IDE tipo Kile, hay un botón para eliminar los archivos auxiliares.
- Si el documento no tiene referencias cruzadas, ni citas bibliográficas, ni ninguna cosa que necesite archivos auxiliares podemos usar `\nofiles` en el preámbulo y solo creará el `pdf`y el `log`. También se creará el `synctex` si compilamos desde un IDE.
- Añadir `--aux-directory=CARPETA` a la orden de compilar. Así insertará todos los archivos auxiliares en la carpeta que le hemos dicho y buscará también ahí cuando los necesite. Lo podemos cambiar fácilmente en el IDE donde aparecen las órdenes.
- Compilar con la opción `--jobname=CARPETA/OUTPUT INPUT.tex`, así guardará el *pdf* y los archivos auxiliares en la carpeta que queramos.
- Usar *Pandoc*. Esto hace cambiar la forma de trabajar radicalmente.

A.1. Referencias

Auxiliary Files en Dickimaw Books

Prevent pdflatex from writing a bunch of files en TeXExchange

I don't want the .aux, .log and .synctex.gz files when using pdflatex en StackOverflow

`latex-mk`

Apéndice B

Enlaces de interés

B.1. Información general

- *LaTeX Resources* y *LaTeX books* en *Dickimaw Books* la página donde Nicola L. C. Talbot recopila lo que escribe sobre LaTeX y otras cosas.
- *The UK list of TeX Frequently Asked Questions*: una recopilación de dudas de LaTeX que cubre todos los aspectos, desde la instalación al formato pasando por los paquetes.
- *The LaTeX project*: información variada sobre LaTeX, noticias y publicaciones.
- *TeX Users Group*: una web de usuarios de LaTeX creada en 1980 con cosas sobre instalación y tipografía.

B.2. Blogs

- *TeX tips*: un blog con truquillos de LaTeX
- *LaTeX Tips*: más trucos, estos especialmente sobre matemáticas.
- *Some TeX Developments*: otro blog, este más avanzado, suele tratar sobre programar y temas complejos.
- *LaTeX* en *High School Math and Chess*: especialmente interesantes la lista de paquetes y cuando prueba diferentes tipos de documentos.

B.3. Cursos

- Curso de la Universidad de Texas
- Curso de la Universidad de Valladolid

B.4. Plantillas

- *LaTeX templates*
- *Overleaf*

B.5. Otros

- Una lista de herramientas para compilar LaTeX online