Introducción a Markdown y Pandoc

Luis Sanjuán

2015

Índice

1. Introducción práctica al uso de Markdown

1.1. Qué es Markdown

No merece la pena entretenerse con cuestiones técnicas y otros detalles. Al final pongo un enlace a lo que considero exposiciones más precisas, así como la documentación oficial.

Para el usuario, Markdown no es más que un conjunto de marcas, de etiquetas, de signos, que se colocan delante o rodeando a un fragmento de texto para indicar su estructura y también su formato.

Así, por ejemplo, si queremos que una línea de un documento que estamos redactando sea el título de una sección, ponemos una marca delante de esa línea:

```
## Sección principal
```

La marca aquí es la doble almohadilla (`##`).

Otro ejemplo, si queremos que un trozo de una línea se destaque (digamos, en negrita) rodeamos ese texto con otra marca.

```
El **piano** es un instrumento.
```

La marca es ahora un doble asterisco rodeando la palabra que queremos destacar, "piano".

Para cada elemento estructural de un documento (encabezados, listas, etc.) hay una marca característica, como también la hay para negritas, cursivas, vínculos a páginas web, etc.

Nuestro primer objetivo es aprender esas marcas básicas y usarlas en la práctica.

Cuando empleamos un procesador de textos como Word u OpenOffice, tenemos que ir por los botones y los menús para conseguir estas mismas cosas. Internamente, el procesador de textos está marcando el documento. La diferencia es que las marcas que pone son demasiado complicadas y por eso proprociona herramientas gráficas para introducirlas (los botones y los menús). Markdown permite olvidarse de levantar la mano del teclado, puesto que la mayoría de sus marcas son sencillas y se aprenden en muy poco tiempo.

La otra ventaja es que existen poderosos conversores de documentos etiquetados con estas marcas de Markdown a otros formatos conocidos, **`pdf`**, **`docx`**, **`html`**, etc. Lo que significa que podemos escribir nuestro documento con las marcas de Markdown y convertirlo a cualquier otro formato muy fácilmente.

1.2. Marcas básicas

La primera parte del trabajo sería conocer y usar las marcas más sencillas, las que se usan en prácticamente todo tipo de documentos. Las listo aquí. Algunas tienen varias posibilidades. Elijo una que me parece más sencilla, por unificar también lo que vayamos a usar todos. Después de listarlas, propongo unas prácticas.

1.2.1. Párrafos

Un párrafo es algo tan basico como dejar una línea en blanco entre un párrafo y el siguiente:

```
Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla.

Y ahora empieza otro párrafo: bla, bla, bla, bla, bla,
bla, bla.
```

1.2.2. Títulos o encabezados de secciones y subsecciones

Para los títulos usamos almohadillas.

El título de una sección principal va con una almohadilla delante:

```
# Título de primer nivel
```

El título de una subsección van con dos almohadillas delante:

```
## Mi titulo de subsección
```

El de una subsubsección con tres:

```
### Título de una subsubsección
```

Y así sucesivamente.

1.2.3. Negritas, cursivas

Las palabras o grupos de palabras en cursiva van rodeados de un asterisco:

```
Hola Hola Hola Hola *adiós adiós*. Los dos adiós en cursiva.
```

Lo mismo para la negrita, pero ahora dos asteriscos:

```
Hola Hola Hola Hola **adiós adiós**. Los dos 'adiós' en negrita.
```

1.2.4. Listas

Las listas numeradas se etiquetan poniendo el número que corresponde a cada item seguido de un punto

```
Lo que sigue será una lista numerada:

1. El primer item de la lista
2. El segundo item
3. El tercero
```

Para listas no numeradas se pone un asterisco delante de cada item, o un guión, o un +. Como el asterisco lo hemos usado para otra cosa, propongo poner un guión:

```
Lo que sigue será una lista no numerada:

- El primer item de la lista
- El segundo
- El tercero
```

1.2.5. Citas

Para citas se usa el ángulo > delante del párrafo que se cita:

```
Como dijo Cervantes:

> En un luguar de la Macha de cuyo nombre no quiero
acordarme ...
```

Si la cita contiene varios párrafos cada párrafo citado debe ir precedido por esa marca:

```
Como dijo Cervantes:

> En un luguar de la Macha de cuyo nombre no quiero
acordarme ...

> En los nidos de antaño no hay pájaros hogaño
```

1.3. Dónde practicar

Para prácticar propongo dos sitios web. Ambos constan de una ventana dividida en dos partes. La de la izquierda es para escribir el texto con las marcas. La de la derecha muestra como aparecería en una página web:

http://dillinger.io/

http://markable.in/editor/

1.4. Ejercicios

1. Probar una a una las marcas citadas en cualquiera de los sitios web que acabo de mencionar (u otros semejantes).
2. Seguir probando (2 min. al día) ...
3. Elegir cualquier texto vuestro, un acta, una página de la programación, lo que sea, y ponerle estas marcas. O bien, crear un documento donde se usen. Propongo que el texto etiquetado así lo adjuntemos a un email para que los demás podemos ver si está bien o sugerir correcciones si es necesario. El asunto del email podría ser "GrupoTrabajo práctica 1".

1.5. Información más detallada sobre Markdown

- Documentación original de Markdown (en inglés): http://daringfireball.net/projects/markdown/
- Página de wikipedia en español sobre Markdown: http://es.wikipedia.org/wiki/Markdown

2. Markdown: Otras marcas interesantes

Conocido ya lo básico, os comento otras marcas útiles, aunque de uso, quizá, menos frecuente.

2.1. Listas anidadas

Una lista anidada es una lista dentro de otra lista. En Markdown, para distinguir la lista interna se utilizan 4 espacios o 1 tabulador delante de cada uno de sus elementos:

```
1. Primer ítem, que contendrá una lista de subapartados:
    - El primer subapartado
    - El segundo
    - El tercero
2. Ahora sigo con otro elemento de la lista principal
3. Y otro
```

2.2. Listas con más de un párrafo en alguno de sus ítems

Aunque tampoco es frecuente, a veces un elemento de una lista consta de varios párrafos. Para que Markdown entienda que no estamos empezando un párrafo nuevo fuera de la lista, sino dentro de un elemento de la lista, se utilizan otros 4 espacios o 1 tabulador delante de ese nuevo párrafo dentro del ítem:

```
1. Esto es un ítem largo. El primer párrafo sería éste.

    Dentro del mismo elemento está este nuevo párrafo.
2. Ahora viene un segundo elemento.
3. Y un tercero.
```

2.3. Hipervínculos (enlaces web)

Existen varias formas, la más sencilla para que aparezca una dirección web o de correo, que se pueda pinchar y llevarnos al sitio correspondiente es rodearla de ángulos.:

```
<http://conservatoriodeavila.es/web>
<usuario@correo.com>
```

2.4. Imágenes

Para incluir imágenes dentro de un documento se utiliza la siguiente sintaxis. Es la más compleja que hemos visto hasta ahora:

```
![título de la imagen](dirección de la imagen)
```

Por ejemplo, si tengo una imagen de un violín (violin.jpg) guardada en la ruta `C:\Fotos\violin.jpg`, para insertarla pondría:

```
![un violín](C:\Fotos\violin.jpg)
```

Lo que va entre corchetes es el título de la imagen, el que queramos nosotros. Lo que va entre paréntesis es la ruta del fichero que contiene la imagen. He puesto una ruta típica en Windows. Para usuarios de MacOSX o Linux, las rutas tienen otra forma, por ejemplo:

```
![un violín](/home/luis/Fotos/violin.jpg)
```

Para ser más breves, podemos suprimir el título y, si la imagen está en la misma carpeta en la que está el documento Markdown, simplificar la ruta. Por ejemplo:

```
![](violin.jpg)
```

¡Ojo! El signo `!` y los corchetes y paréntesis son obligatorios.

2.5. Tablas

Hay muchas opciones para hacer tablas, pero, en general, son engorrosas. La más sencilla tendría una forma parecida a esta:

```
Alumno    Nota    Faltas
------    ----    ------
Pepita    10      0
Juanín    0       10
```

Los encabezados van arriba separados de las filas de las tablas por sucesiones de guiones. Luego van en lineas distintas cada una de las filas. Alguien pude estar interesado en probar cómo afecta la alineación de los items en cada celda con respecto a la linea de guiones. En el ejemplo, esos elementos están alineados con el comienzo de la línea de guiones para cada columna. ¿Qué pasaría si estuvieran alineados con el final de la línea de guiones en cada columna, o no alineados ni a la izquierda ni a la derecha?

Sin embargo, esta sintaxis sólo funcionaría con Pandoc u otros interpretes online de Markdown que la implementen. En el Markdown original no hay soporte para tablas. Dicho soporte, junto con otros como el soporte para ecuaciones matemáticas, es una extensión al Markdown original, y depende de los intérpretes.

http://Dillinger.io y muchos de los intérpretes online, aceptan esta otra sintaxis, algo más engorrosa:

```
|Alumno|Nota|Faltas|
|------|----|------|
|Pepita|10  |0     |
|Juanín|0   |10    |
```

(Nota: Los separadores `|` no tienen porque estar alineados. Lo pongo así porque es más fácil de ver para el ojo humano.)

2.6. Un par de extras

Dos marcas muy usuales en documentos técnicos son: texto *verbatim* (que aparecerá con tipo de letra *typewritter* y donde el espaciado original, el que pone el que lo escribe, se respeta) y ecuaciones matemáticas.

El texto *verbatim* es el que uso yo para poner los ejemplos de cada marca. Y se consigue poniendo cuatro espacios delante de cada línea de este texto. Por ejemplo:

```
Aquí estoy en el texto normal, pero el siguiente párrafo va a contener
un ejemplo de uso de Markdown y lo voy a desplazar cuatro espacios
respecto de este texto normal:

    <http://www.example.com>
    <http://conservatoriodeavila.es/web>
```

Termino con una ecuación matemática. ¿Qué tal la fórmula de la media? Esto no funciona habitualmente en intérpretes online, salvo aquellos que además soportan MathJax. Habrá que esperar a pandoc para ver el resultado.

```
$\mu = \frac{1}{n}\sum_{i=1}^n a_i$
```

2.7. Ejercicios

1. Probar independientemente cada una de las marcas explicadas. Tened en cuenta que las siguientes seguramente no funcionen en http://Dillinger.io u otros intérpretes online de Markdown como los que indiqué en la primera entrega:

   - Imágenes. Si el sitio en concreto no permite cargar imágenes, y es improbable que lo haga, esto no funcionará.

   - Tablas. Sólo funcionará (en http://Dillinger.io) la segunda comentada.
   - Ecuaciones. No funciona salvo que el sitio soporte MathJax (improbable).

2. Aún sin comprobar online (esperemos a Pandoc), crear un mini-documento donde se incluyan listas anidadas, hipervínculos e imágenes (ej. el logo del conser). Y opcionalmente, tablas.

3. Instalación de Pandoc

Pandoc es la utilidad más completa y potente para convertir cualquier tipo de documento en cualquier otro tipo de documento. Es especialmente apropiada para convertir de Markdown a otros múltiples formatos.

Naturalmente, el primer paso es instalar Pandoc. Pero para poder usar la conversión a `pdf`, tendremos que instalar también una distribución de TEX.

3.1. TEX, la máquina tipográfica

Dicho muy sumariamente, llamo distribución de TEX al conjunto de paquetes y utilidades en torno a la máquina de tipografía digital conocida en términos genéricos como TEX. Inventada por D. Knuth, uno de los padres de la teoría de algoritmos. En la actualidad, la máquina que se usa es una variante de TEX, a saber, pdfTEX. Pero se acostumbra a utilizar el término TEX para todas estas herramientas de tipografía digital cuyo origen es la obra de Knuth. Estas herramientas son, hoy por hoy, y desde hace ya mucho tiempo, la opción profesional más potente que existe, y es la que utilizan las editoriales más prestigiosas del mundo para producir sus libros y revistas científicas.

Hay muchas distribuciones de TEX, varias según el sistema operativo. Yo uso Linux y sólo tengo experiencia en instalar TEX en Linux. Lo único que puedo en

este sentido es indicar los sitios donde están las distribuciones más conocidas para Windows y MacOSX. Si alguno tiene Linux, añadiré más tarde cómo se instala en Linux.

La distribución completa de TEX es un programa muy grande, en torno a 2GB. Una instalación completa permite olvidarse de todo, todas las fuentes estarán instaladas, todos los paquetes. Pero, dado el tamaño, normalmente se opta por instalar una base a la que se van añadiendo paquetes según se necesitan.

Las distribuciones más usadas son las siguientes. Las instrucciones de instalación aparecen en sendas páginas:

- Para Windows: MikTEX (http://miktex.org/download)
- Para MacOSX: MacTEX (http://www.tug.org/mactex/morepackages.html)

3.2. Pandoc, el conversor

Una vez instalado TEX, la instalación de Pandoc no debería entrañar ningún problema. En enlace está aquí. En la parte inferior de la página aparecen botones para cada sistema operativo.

https://github.com/jgm/pandoc/releases

Las instrucciones de instalación, que indican, sumariamente, lo que he comentado antes están aquí:

http://johnmacfarlane.net/pandoc/installing.html

3.3. Terminal, la interfaz de usuario

Pandoc se utiliza sobre el terminal. No hay que tener miedo. El terminal es mucho más potente y flexible, y para el uso que le vamos a dar, es cosa de aprender unas pocas órdenes.

Una primera toma de contacto con el terminal nos servirá para confirmar que Pandoc se ha instalado correctamente.

Para eso hay que seguir las instrucciones indicadas en el punto *Step2* de esta página:

http://johnmacfarlane.net/pandoc/getting-started.html

Hay instrucciones para cada sistema operativo. Si encontráis dificultades con el inglés, me lo decís y lo traduzco.

3.4. Ejercicios

El ejercicio esta vez es muy simple: proceder a la instalación de TEX y Pandoc y pasar por las instrucciones comentadas en la sección anterior. Como confirmación de que hemos seguido el proceso, propongo que, quien quiera, envié un post al foro donde aparezca lo que resulta de ejecutar en el terminal estas dos órdenes:

```
pdftex --version
```

y, tras copiar el resultado para postearlo en el foro,

```
pandoc --version
```

Son dos órdenes independientes. Por cierto, a este tipo de órdenes se les llama también *comandos* (spanglish del original inglés *command*)

4. Introducción a Pandoc

Pandoc es un programa que convierte documentos de un formato a otro. La potencia de Pandoc es muy grande. Nosotros sólo vamos a utilizarlo a una pequeña escala, la necesaria para nuestros propósitos.

Pero bueno es conocer que las posibilidades son bastantes más de las que vamos a ver. En concreto, mirad el comienzo de la documentación aquí:

http://johnmacfarlane.net/pandoc/README.html#general-options

Se especifican los formatos `from` y `to` soportados. `from` indica el formato de origen del documento y `to` el de salida. Por ejemplo, Pandoc puede convertir un documento **`epub`** (típico de ebooks) a **`beamer`** (típico de diapositivas para conferencias).

Para nuestros propósitos nos limitaremos a documentos **`markdown`** (con extensión **`md`**, como formato de entrada y a **`pdf`**, **`odt`** y **`docx`** como formatos de salida.

4.1. La línea de comandos

Lo primero es entrar en la línea de comandos (es spanglish, siendo correctos sería línea de órdenes, pero el uso dominante es *comandos*). Tendréis que mirar en vuestro correspondiente sistema operativo la forma de entrar en la línea de comandos. Si hay dudas en esto, el foro está ahí.

Una vez en la línea de comandos, desplazáos a la carpeta (= directorio) donde tenéis vuestros documentos Markdown, las prácticas que hemos hecho hasta ahora. Suponiendo que están en el directorio *GrupoTrabajo* dentro de la carpeta *Documentos* tendréis que hacer algo así (las rutas son aproximadas, ya que dependen de cada sistema operativo):

- En Windows:

  ```
  cd C:\Users\Documentos\GrupoTrabajo
  ```

- En MacOSX:

  ```
  cd /Users/Documentos/GrupoTrabajo
  ```

- En Linux:

  Si alguno tiene Linux ya sabrá cómo hacerlo y dónde ir.

4.2. Ejecutar Pandoc

Como esto es una práctica donde hay que utilizar la línea de comandos, aparte de ejecutar Pandoc, nos limitaremos a lo más simple: ver cómo ejecutar el programa de la forma más elemental posible y ver cuál es el resultado.

Escoged una de vuestras prácticas, mejor la más breve posible. Digamos que el fichero de esa práctica es **`practica1.md`**. Si no las habéis guardado en ningún fichero hacedlo ahora. No olvidad terminar el fichero con la extensión **`md`**. El punto marca el comienzo de la extensión y no hay espacio entre el nombre, el punto y la extensión. En general conviene que los nombres de ficheros no contengan espacio alguno ni caracteres raros. Como separador, si el nombre del fichero contiene varias palabras, es útil el guión bajo. Por ejemplo: **`practica_1.md`**.

La ejecución de Pandoc sobre este fichero es muy simple:

```
pandoc practica1.md
```

Por defecto, `pandoc` produce el resultado y lo muestra en pantalla. El formato de conversión por defecto es **`html`**.

Terminemos añadiendo una opción a la ejecución, la opción `--standalone`, que genera un documento autónomo. Veremos qué significa tal cosa en breve.

Las opciones van justo después de `pandoc` y separadas por un espacio. Típicamente, se proporcionan dos formas equivalentes para la mayoria de las opciones, una con formato largo y una abreviada. Las primeras van precedidas por dos guiones; las segundas por uno solo. Más fácil es verlo en acción. Ejecutad:

```
pandoc --standalone practica1.md
```

Y con la forma abreviada:

```
pandoc -s practica1.md
```

El resultado es el mismo.

4.3. Ejercicios

Como prueba de que habéis realizado con éxito esta practica, enviad al foro el resultado de la última ejecución. Normalmente se puede cortar y pegar desde la línea de órdenes.

Finalmente, ¿por qué lo de *autónomo*? La respuesta en próximas prácticas. Pero si alguien no puede esperar que ejecute:

```
pandoc -s -o practica1.html practica1.md
```

y visite el recién creado documento **`practica1.html`** desde su navegador web.

5. Conversión de documentos con **Pandoc**

La práctica de hoy es la más sencilla, la más corta, y a la vez, la más gratificante. Se trata de obtener distintos documentos finales (en diversos formatos) a partir de un documento Markdown.

En definitiva, lo que conseguimos de este modo es atenernos a aquel ideal comentado en el foro. El escritor escribe su texto y lo marca para que el tipógrafo produzca el resultado deseado. El escritor debe desentenderse lo más posible de las cuestiones de formateo y diseño. Idealmente, las marcas deben ser muy ligeras. Markdown es el lenguaje de marcas más ligero que existe hasta la fecha y de ahí su conveniencia y su muy amplia difusión. El resultado, desde el punto de vista tipográfico, será de mayor o menor calidad dependiendo del tipógrafo, claro. Puesto que el tipógrafo más experto es TEX, el mejor resultado lo obtendremos al producir un documento **`pdf`**, que es la salida que produce TEX (concretamente, pdfTEX, su moderno sucesor). Los resultados en **`odt`**, que es un formato estándar (*Open Document Text*) para documentos de texto, el nativo de OpenOffice y LibreOffice, así como **`docx`**, que es el formato para documentos de texto de Microsoft Office actual, son más rupestres. Al fin y al cabo en los procesadores de texto el escritor también tiene que encargarse del resultado, diseño y tipografía, y lo que obtenemos en nuestra conversión será el resultado tal como queda definido por la plantilla básica que Jonh MacFarlane (el creador de Pandoc) ha diseñado.

La otra gran ventaja, y que conecta directamente con el título de este curso, es que es infinitamente más fácil editar conjuntamente un texto plano, como es un texto etiquetado con Markdown, que editar un texto con formato, incluidos formatos indicados para esa función, como **`odt`** y **`docx`** (**`pdf`** no fue diseñado para ser editado). Todos sabemos que cambiar un documento en un formato de procesador de texto puede ser problemático. Que si la fuente se cambia, que si la imagen se va al garete, que si el párrafo se desalinea, etc. Con un texto plano nada de esto sucede.

5.1. Sintaxis de los comandos

Los comandos, las órdenes, son muy simples. Nos atenemos a lo básico. Todos estos comandos incluyen la opción `--standalone` o `-s` (su forma abreviada). De esta forma obtenemos un documento independiente, en lugar de una parte de un documento para ser incluida en un documento maestro. Todos los comandos tiene la misma sintaxis:

```
pandoc --standalone --output mi_documento.<formato_de_salida> mi_documento.md
```

O en su forma abreviada:

```
pandoc -s -o mi_documento.<formato_de_salida> mi_documento.md
```

En estos comandos `mi_documento.<formato_de_salida>` es el nombre del fichero de salida que quiero obtener, donde formato de salida es la extensión

que corresponde a dicho formato. Esto es, si quiero obtener un documento **pdf** será **mi_documento.pdf**; si quiero un documento **docx** será **mi_documento.docx**. Naturalmente, en lugar de **mi_documento**, utilizo el titulo de fichero que deseo. Digamos **memoria.pdf** o **acta.docx**, etc. No hace falta que coincidan el nombre de fichero de entrada y el de salida, aunque es normal que lo hagan. Es decisión del usuario, en todo caso.

5.2. Los comandos concretos

Así, por tanto, estos serían los comandos para obtener los tres tipos de documentos comentados:

- Documento **pdf**

  ```
  pandoc -s -o mi_documento.pdf mi_documento.md
  ```

- Documento **odt**

  ```
  pandoc -s -o mi_documento.odt mi_documento.md
  ```

- Documento **docx**

  ```
  pandoc -s -o mi_documento.docx mi_documento.md
  ```

5.3. Ejercicios

Simplemente poned en el foro el **pdf** y el **docx** (o el **odt**) que resulte de procesar la práctica más compleja que hayáis hecho hasta ahora. De esta manera sabremos si todas las marcas (enlaces, imágenes, etc.) funcionan correctamente.

6. Pandoc Avanzado: LaTeX

Titulo *Pandoc Avanzado* esta práctica. Pero no lo es. Sería más bien Pandoc a nivel intermedio.

6.1. El verdadero proceso de conversión

La conversión que realiza Pandoc a la hora de obtener un **pdf** se realiza en realidad a través de LaTeX, solo que es transparente al usuario, a no ser que el usuario solicite que no lo sea.

El proceso interno es, más bien, una conversión de **markdown** a LaTeX y, después, la creación de un **pdf** a partir del fichero LaTeX intermedio [Estos ficheros tienen la extensión **tex**]. Dicha creación del **pdf** es obra de TeX, o más exactamente de su más moderno y popular descendiente pdfTeX:

```
             (pandoc)               (pdflatex)
documento.md -------> documento.tex ---------> documento.pdf
```

Si queremos, podemos seguir estos mismos pasos nosotros mismos:

```
pandoc -o mi_practica.tex mi_practica.md
pdflatex mi_practica.tex
```

El último comando genera el fichero **mi_practica.pdf**.

La utilidad del paso intermedio la veremos en otra práctica, más bien demostración, ya que eso sí sería Pandoc avanzado. Pero la redactaré hacia el final de esta serie para mostrar la flexibilidad de estas herramientas, abriendo quizá la posibilidad de que podáis investigar en el futuro por vuestra cuenta.

6.2. Plantillas

Cierto grado de flexibilidad para el usuario de a pie es en todo caso posible gracias a lo que Pandoc llama plantillas.

Pandoc proporciona una plantilla de LaTeX por defecto, que, por si alguien quiere mirarla (con un editor de texto plano), está en:

```
$PANDOC_DIR/data/templates/default.latex
```

(O con barras invertidas \ en Windows)

Lo que a efectos prácticos esta plantilla permite es definir, mediante lo que en terminología Pandoc se llama *campo de metadatos* (*meta-data field*), ciertos valores variables del documento, como el título, la fecha, el autor, el tamaño del papel, el tamaño de la fuente, y unos pocos más.

Lo interesante es que LaTeX por defecto proporciona estas mismas variables y actúa adecuadamente ante ellas.

Sin más palabras. La práctica de hoy sería sencilla. Añadir al principio de una de vuestras prácticas previas un campo de metadatos como el siguiente [tres guiones arriba y abajo y las claves en inglés seguidas de dos puntos]:

```
---
lang: spanish
title: Titulo de la práctica
author: Vuestro nombre
date: la fecha que queráis
fontsize: 12pt
---
```

y generar el **pdf** como vimos el día pasado.

Por cierto, y puestos ya a experimentar, incluso os puede resultar interesante encadenar varias de las prácticas como subpartes del mismo documento. La primera es la que contendría el campo de metadatos.

El comando sería:

```
pandoc -s -o todas_las_practicas.pdf practica1.md practica2.md practica3.md
```

las que sean, las que tengáis o queráis encadenar.

Esto último, solicitar la creación de un único **pdf** a partir de varios ficheros **md** es realmente útil, ya que podemos dividir un trabajo grande en partes más pequeñas mucho más fáciles de manejar.

6.3. Ejercicios

Adjuntad un resultado como prueba de que todo sale como se espera. Como queráis, una sola práctica con su campo de metadatos y el resultado en **pdf**, o un único **pdf** derivado de múltiples prácticas, la primera de las cuales contenga el campo de metadatos propuesto.

7. Documentos de referencia

Del mismo modo que es posible personalizar con bloques de metadatos la salida a **pdf**, es posible también personalizar la salida a **odt** (formato de OpenOffice o LibreOffice) o **docx** (formato de Microsoft Office).

[En lo que sigue me referiré a documentos de Microsoft Office (**docx**), porque es lo que usáis, según creo. Pero funciona igual para los documentos de OpenOffice o LibreOffice (**odt**).]

En el caso de este tipo de documentos la forma habitual de personalizar es lo que Pandoc denomina una *reference*, que es simplemente un documento de referencia en formato **docx** al que se le han aplicado, mediante las correspondientes herramientas gráficas de los procesadores de textos, los estilos que interese mantener en nuestro documento de salida.

La idea es simple:

1. Creo un documento en Office, puramente esquemático, pero que contenga todos los elementos estructurales que contendrá el verdadero documento Markdown que voy a convertir a **docx**.
2. Aplico los estilos que me interesen a cada uno de esos elementos estructurales.
3. Ejecuto `pandoc` para que convierta a **docx** mi documento en formato **md**, pero indicándole que tome como referencia ese otro documento esquemático al que he aplicado ya los estilos que me interesa reproducir.

En principio, nada impide que el documento de referencia, en lugar de ser un documento real, aunque muy esquemático, sea un documento vacío, sobre el que se han definido los estilos con la herramienta correspondiente del procesador de textos, algo en la línea de lo que se explica en este blog:

http://hackademic.postach.io/pandoc-and-academic-docx-files

No puedo poner un ejemplo concreto con Office, porque no lo tengo instalado. Si me insistís puedo poner un ejemplo con LibreOffice, que tampoco uso, pero que sí tengo instalado. Pienso que de todas formas la idea es bastante simple y podéis ser vosotros mismos quienes experimentéis.

7.1. El comando

Falta decir el comando que requiere Pandoc para conocer nuestro documento de referencia. Supongamos que el documento de referencia lo hemos guardado con el nombre **`plantilla.docx`** y que nuestro documento es **`mi_documento.md`**. El comando para convertir a **`docx`** siguiendo los estilos de referencia en **`plantilla.docx`** sería:

```
pandoc -s -o mi_documento.docx --reference-docx plantilla.docx mi_documento.md
```

Por si sirve para aclarar la sintaxis, una transcripción a nuestra forma más expresiva de hablar sería:

> Pandoc, genera un documento autónomo (opción `-s`) a partir de **`mi_documento.md`**, cuya salida (opción `-o`) sea **`mi_documento.docx`**, y que tome como referencia de estilos (opción `--reference-docx`) el fichero **`plantilla.docx`**.

Notad que `--reference-docx` lleva dos guiones al principio, pues es una opción en formato largo.

7.2. LaTeX *versus* procesadores de texto

Uno de los problemas de esta estrategia es que los nombres de los estilos en los procesadores de texto pueden cambiar de versión en versión, y por supuesto de un procesador a otro. Por no hablar de las fuentes. Según el sistema operativo, unas fuentes pueden estar instaladas o no. Cuando un sistema no dispone de la fuente, los procesadores de texto declaran fuentes virtuales, Así, por ejemplo, *Arial* no está disponible en Linux por defecto y mi LibreOffice la declara como *Arial* virtual. La conversión de un *Arial* virtual a una fuente real en mi sistema operativo no va a tener el mismo aspecto que el *Arial* de Microsoft.

Todos estos problemas de incompatibilidad entre sistemas operativos, distintos programas o, incluso, distintas versiones del mismo programa, son prácticamente inexistentes en el caso de LaTeX, puesto que es igual para todos los sistemas operativos y no ha variado en el soporte esencial a lo largo de muchos años.

7.3. El ciclo de trabajo ideal

Termino hoy con una reflexión sobre lo que, en mi opinión, sería el proceder ideal en la generación de documentos oficiales que han de pasar por varias manos y revisiones, al menos la mayoría de ellos, por ejemplo, programaciones y memorias.

El mejor formato de salida posible, por su perfección tipográfica y su coherencia es **pdf**. No hay nada mejor para ser impreso, y con fuentes adecuadas, para ser visto en pantalla de ordenador. Si se requiriese un documento para ser visto en cualquier tipo de pantalla (tablet, móvil, etc.) **epub** sería seguramente la mejor opción (Pandoc soporta conversión a **epub**).

La edición conjunta se realizaría sobre el documento **markdown**. Texto plano que no da problemas y es fácil de editar, modificar, etc.

Una vez obtenida una versión final, se procesaría con Pandoc para convertirlo a **pdf**. Si se requiriesen, por lo que fuera, cosas especiales, se podrían crear plantillas especiales en lugar de la plantilla LaTeX por defecto de Pandoc.

Si, en todo caso, fuese necesario distribuir un **docx**, habría que crear documentos de referencia para cada tipo de documento. En todo caso la necesidad de obtener **docx** queda ya muy limitada, pues las virtudes de la flexibilidad de la edición ya las proporciona Markdown.

Naturalmente, es sólo un ideal. No creo que, hoy por hoy, hubiese disposición general de aprender Markdown. Y la creación de plantillas lleva, por supuesto, un buen trabajo.

Pero hablar de lo ideal creo que no viene mal.

En otro ámbitos lo ideal se hace real a base de ser un requisito insoslayable ;-) Por ejemplo, es típico en el ámbito de los analistas de datos divulgar sus análisis etiquetados con Markdown y procesarlos con KnitR, que usa Pandoc por detrás. O si nos vamos a editoriales científicas, es típico que los autores tengan que enviar sus obras etiquetadas directamente con LaTeX.

7.4. Ejercicios

Experimentar con la creación de un documento de referencia y procesad una de vuestras prácticas anteriores, o una nueva, en la forma indicada.

8. Pandoc realmente avanzado: plantillas LaTeX

Si no surge la necesidad de añadir más materiales, me parece bien terminar nuestro encuentro con Pandoc con un ejemplo de uso, esta vez, sí, realmente avanzado.

Se trata de mostrar la flexibilidad de las herramientas y de, una vez que logramos dominar en alguna medida esa flexibilidad, saber cómo sacar provecho de ella para adaptarla a nuestras necesidades y automatizar la parte engorrosa y repetitiva de la escritura.

Esta sección va a ser más larga. Empiezo ahora a redactarla y no sé si dividirla en varias entregas. Iré viendo por el camino. El objetivo es ver lo que es posible. No tengo idea ahora mismo de proponer alguna práctica sobre ella. Su propósito es ilustrativo, en el sentido de que si se conoce lo que es posible, quizá algún día, con más tiempo o conocimientos, cada cual puede seguir la misma o parecida estrategia.

8.1. El objetivo concreto del ejemplo

Nuestro objetivo particular va a ser producir un acta de departamento en `pdf` siguiendo el modelo propuesto. No seguiré la geometría exacta del modelo, aunque es perfectamente posible si personalizamos la geometría de la página de una forma parecida a como haremos con otras variables. No merece la pena complicar la exposición con este detalle, por lo demás no significativo, puesto que dicha geometría puede ser perfectamente otra, y si es la que es, supongo que es más bien por azar y no porque sea oficialmente prescrita.

En la página siguiente tenéis una muestra de un acta que sigue el modelo del conservatorio, tal y como se presentan en la actualidad.

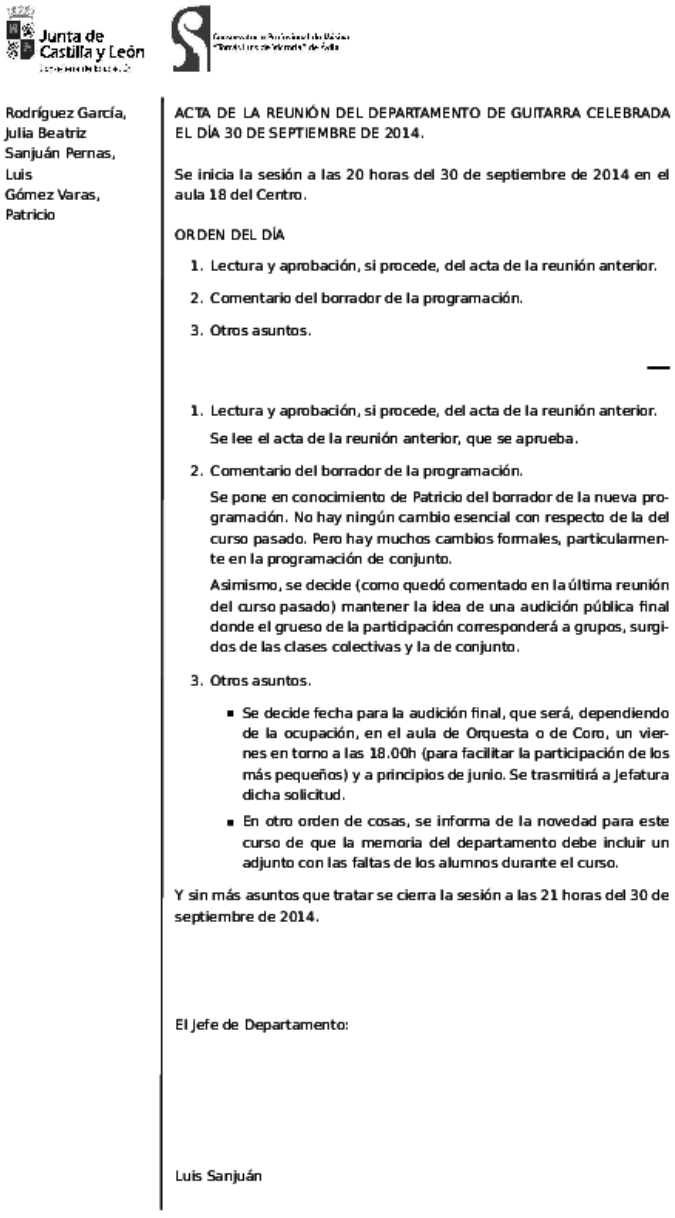
8.2. Presupuestos iniciales

Antes de plantearnos la forma de atacar el problema es conveniente recordar lo que ya sabemos acerca de Pandoc.

- Pandoc es un conversor de propósito general que ofrece plantillas por defecto para convertir a diversos formatos.
- Para la conversión a `pdf` Pandoc proporciona una plantilla LaTeX por defecto.
- Ciertos elementos en dicha plantilla son variables (como el título, el autor, la fecha, etc) y se pueden definir para cada documento concreto mediante *bloques de metadatos*.
- Cualquiera de nuestras prácticas anteriores muestra con evidencia que la plantilla por defecto no es capaz de algo tan específico como dos columnas separadas por una línea. Tampoco queda claro hasta qué punto será complicado insertar la marca de aguas con el logo tan pegada al margen superior. Desde luego, nada de esto resulta obvio.

8.3. Posibles líneas de ataque y posibilidades que investigar

Dados los presupuestos anteriores podemos plantearnos las siguientes posibilidades, algunas expresadas en forma de preguntas o de meras tentativas.



Junta de Castilla y León

Rodríguez García, Julia Beatriz
Sanjuán Pernas, Luis
Gómez Varas, Patricio

ACTA DE LA REUNIÓN DEL DEPARTAMENTO DE GUITARRA CELEBRADA EL DÍA 30 DE SEPTIEMBRE DE 2014.

Se inicia la sesión a las 20 horas del 30 de septiembre de 2014 en el aula 18 del Centro.

ORDEN DEL DÍA

1. Lectura y aprobación, si procede, del acta de la reunión anterior.
2. Comentario del borrador de la programación.
3. Otros asuntos.

—

1. Lectura y aprobación, si procede, del acta de la reunión anterior.

   Se lee el acta de la reunión anterior, que se aprueba.

2. Comentario del borrador de la programación.

   Se pone en conocimiento de Patricio del borrador de la nueva programación. No hay ningún cambio esencial con respecto de la del curso pasado. Pero hay muchos cambios formales, particularmente en la programación de conjunto.

   Asimismo, se decide (como quedó comentado en la última reunión del curso pasado) mantener la idea de una audición pública final donde el grueso de la participación corresponderá a grupos, surgidos de las clases colectivas y la de conjunto.

3. Otros asuntos.

   - Se decide fecha para la audición final, que será, dependiendo de la ocupación, en el aula de Orquesta o de Coro, un viernes en torno a las 18.00h (para facilitar la participación de los más pequeños) y a principios de junio. Se trasmitirá a Jefatura dicha solicitud.
   - En otro orden de cosas, se informa de la novedad para este curso de que la memoria del departamento debe incluir un adjunto con las faltas de los alumnos durante el curso.

Y sin más asuntos que tratar se cierra la sesión a las 21 horas del 30 de septiembre de 2014.

El Jefe de Departamento:

Luis Sanjuán

Figura 1: modelo de acta

1. Teóricamente es posible utilizar plantillas personalizadas en lugar de la que Pandoc proporciona por defecto. ¿Hasta qué punto es posible? En concreto, ¿es posible, y cómo, crear una plantilla de LaTeX para lograr emular el modelo de acta referido?

2. Sabemos que podemos definir variables en campos de metadatos, pero ¿qué pasa con variables que no están descritas en las que Pandoc proporciona como tales? ¿Es posible crear variables nuevas, tales como *profesor*, *hora* de reunión, etc.

3. Acerca de la marca de agua del logo. Parece que habría dos opciones:

   - Modificar la geometría de la página para que la imagen del logo apareciera pegada al borde superior.
   - Investigar el asunto de marcas de aguas. Parece lógico pensar que algo tan común como una marca de agua debe de tener alguna respuesta relativamente sencilla.

8.4. Plantillas personalizadas para LaTeX

Procedamos una por una tratando de responder y experimentar acerca de las líneas de actuación y preguntas antes formuladas.

En efecto, no sólo es teóricamente posible, sino realmente factible crear plantillas personalizadas de LaTeX para Pandoc, ya sea adaptando la plantilla por defecto, ya creando una enteramente nueva.

Como este es un documento muy particular, casi ninguna de las variables que aparecen en la plantilla por defecto de Pandoc nos interesan. Tampoco nos interesan la gran cantidad de opciones que hay ahí presentes. Ciertamente, la plantilla por defecto de LaTeX que Pandoc suministra es muy compleja, con el fin de atender a múltiples variantes de documentos de propósito general. En definitiva, para hacer las cosas simples, en otras palabras, para reducir al máximo el tamaño de nuestra plantilla, parece razonable crear una desde cero.

Esta plantilla deberá contener todo lo necesario, y sólo lo necesario, para reproducir el modelo.

Naturalmente, la plantilla no es otra cosa que un documento escrito con instrucciones y etiquetas de LaTeX. No me voy a detener en LaTeX, pues no es el propósito de este curso. Además, un conocimiento suficiente de LaTeX como para crear un documento de esta clase incluye saber LaTeX a un nivel como mínimo intermedio, así como tener soltura a la hora de descubrir las extensiones (*paquetes* en términos de LaTeX) que nos serán de utilidad. Lo primero exige al menos un par de cursos específicos con la duración prevista para éste. Lo segundo es cosa de experiencia. CTAN, el repositorio de paquetes de LaTeX, contiene miles de extensiones, con las que es posible hacer lo imaginable y lo inimaginable a nivel tipográfico y de diseño, desde escribir en toda clase de lenguas, escritura fonética, escritura musical, cientos de cosas para documentos científicos, e incontables posibilidades para la creación de diseños. En tal océano de paquetes no es fácil dar exactamente con el que nos interesa para cada situación concreta fuera de lo

común, a no ser que sepamos de su existencia o búsquedas en Google u otros motores nos ayuden.

El problema del caso tiene diversas formas posibles de solución, desde construir alguna clase de tabla con dos celdas, pasando por crear un documento con dos columnas de tamaño definido por el usuario, hasta usar la opción de notas al margen: una nota en el margen izquierdo contendría la lista de profesores asistentes a la reunión. Esta última posibilidad es la que voy a explorar.

El paquete de LaTeX `marginnote` nos permitirá definir exactamente la dimensión y ubicación de la nota al margen.

Para la línea divisoria recurriré al paquete `background`, que permite incluir material de fondo en un documento con un control fino de los detalles de contenido, ubicación, etc.

Incluyo el código LaTeX para ambos propósitos sin más comentario. Tomadlo tal cual; simplemente funciona:

```
% Linea de separación
\SetBgScale{1}
\SetBgColor{black}
\SetBgAngle{0}
\SetBgHshift{-0.52\textwidth}
\SetBgVshift{-1mm}
\SetBgContents{\rule{0.4pt}{\textheight}}

% Definiciones relativas a la nota al margen
\setlength{\marginparwidth}{35mm}
\setlength{\parindent}{0pt}
\renewcommand*{\raggedleftmarginnote}{}
\reversemarginpar
```

Estas personalizaciones irán en lo que en terminología LaTeX se denomina el *preámbulo* del documento, un apartado anterior al cuerpo del documento propiamente dicho y que afecta a su presentación y diseño. En este preámbulo se definen también otras características generales, como el tipo de documento, la lengua en que esta escrito, las fuentes que usar, o ciertas dimensiones globales como el grado de indentación en las primeras líneas de párrafos. Es, en general, el lugar donde se recopilan todas las personalizaciones que afectan a elementos estructurales del documento. De hecho, también he añadido estilos para los títulos de las secciones del acta: la cabecera que, de alguna forma, ocupa el lugar del título del acta, o su sección principal (marca `#` en Markdown) y la subsección correspondiente al orden del día (marca `##` en Markdown). Finalmente, es el preámbulo donde se *cargan* los paquetes que extienden LaTeX más allá de su funcionalidad básica.

Nuestro preámbulo completo hasta aquí sería el siguiente:

```
\documentclass[a4paper]{extreport}
\usepackage[T1]{fontenc}
```

```
\usepackage[utf8]{inputenc}
\usepackage[spanish]{babel}
\usepackage{titlesec}
\usepackage{marginnote}
\usepackage{background}

\setlength{\parindent}{0pt}

% Línea de separación
\SetBgScale{1}
\SetBgColor{black}
\SetBgAngle{0}
\SetBgHshift{-0.52\textwidth}
\SetBgVshift{-1mm}
\SetBgContents{\rule{0.4pt}{\textheight}}

% Definiciones relativas a la nota al margen
\setlength{\marginparwidth}{35mm}
\renewcommand*{\raggedleftmarginnote}{}
\reversemarginpar

% Títulos de secciones
\titleformat{\section}[hang]{\small\bfseries}{}{0pt}{\raggedright\uppercase}
\titleformat{\subsection}[hang]{\small\bfseries}{}{0pt}{\uppercase}
\titlespacing{\section}{0pt}{-10pt}{10pt}
\titlespacing{\subsection}{0pt}{10pt}{10pt}
```

Además del preámbulo, y después de él, una plantilla LaTeX para Pandoc, y en general todo documento escrito en LaTeX, espera lo que se denomina el *entorno del documento* que es donde va su texto (o *cuerpo*) propiamente dicho. Por añadidura, una plantillas Pandoc (esto sí es específico de Pandoc) necesita incluir ahí también la variable `$body$`, que cuando procesemos nuestro documento será sustituida por el contenido que haya en él.

```
\begin{document}

$body$

\end{document}
```

En nuestro caso particular hay que añadir la comentada nota al margen con la lista de profesores asistentes. En consecuencia, incluida la instrucción LaTeX para crear esa nota al margen, el entorno del documento en nuestra plantilla queda así:

```
\begin{document}
\marginnote{\small\mbox{}Gómez Varas\\Patricio\\
                         Rodríguez García\\Julia Beatríz\\
                         Sanjuán Pernas\\Luis}
```

```
$body$

\end{document}
```

Toca procesar el acta, escrita en Markdown, con `pandoc` haciéndole saber que, en lugar de su plantilla por defecto, queremos usar nuestra plantilla. Para indicar la plantilla que aplicar, se utiliza la opción

```
--template <nombre_plantilla>.latex
```

Guardemos nuestra plantilla con el nombre **`plantilla_acta.latex`** y supongamos, además, que tenemos escrita ya el acta en el fichero **`acta.md`**. El acta tiene este aspecto ya familiar:

```
%%% acta_v1.md
# Acta de la reunión del departamento de guitarra celebrada el día 30 de septiembre de 201

Se inicia la sesión a las 20:00 horas del 30 de septiembre
de 2014 en el aula 18 del centro.

## Orden del día

1.  Lectura y aprobación, si procede, del acta de la reunión anterior.
2.  Comentario del borrador de la programación.
3.  Otros asuntos


1.  Lectura y aprobación, si procede, del acta de la reunión anterior.

    Se lee el acta de la reunión anterior, que se aprueba.

2.  Comentario del borrador de la programación.

    Se pone en conocimiento de Patricio del borrador de la nueva programación.
    No hay ningún cambio esencial con respecto de la del curso pasado.
    Pero hay muchos cambios formales, particularmente en la programación
    de conjunto.

    Asimismo, se decide (como quedó comentado en la última reunión del
    curso pasado) mantener la idea de una audición pública final donde el
    grueso de la participación corresponderá a grupos, surgidos de las
    clases colectivas y la de conjunto.

3.  Otros asuntos.

    - Se decide fecha para la audición final, que será, dependiendo de
      la ocupación, en el aula de Orquesta o de Coro, un viernes en torno
      a las 18.00h (para facilitar la participación de los más pequeños) y a
```

```
      principios de junio. Se trasmitirá a Jefatura dicha solicitud.

    - En otro orden de cosas, se informa de la novedad para este curso de
      que la memoria del departamento debe incluir un adjunto con las faltas
      de los alumnos durante el curso.


Y sin más asuntos que tratar se cierra la sesión a las 21:00 horas del 30
de septiembre de 2014.

El Jefe del Departamento:

Luis Sanjuán

%% fin del acta
```

Tenemos todas las piezas preparadas, el acta con nombre **acta_v1.md** y la plantilla LaTeX, con nombre **plantilla_acta.latex**.

Ejecutemos, pues, `pandoc` como otras veces, pero ahora añadiendo la opción `--template` que acabo de comentar:

```
pandoc -s --template plantilla_acta.latex -o acta_v1.pdf acta.md
```

Una imagen del **pdf** resultante se muestra en la página siguiente.

¡Estupendo! Esto se aproxima bastante al modelo. Pero hay todavía algunos problemas imprevistos.

El más grave corresponde a las numeraciones de las listas. La lista del orden del día consta de tres elementos; la correspondiente lista del comentario sobre esos elementos consta también de tres, pero debería haber comenzado con 1, en lugar de continuar la numeración de la lista anterior. La razón de este problema es que la especificación de Markdown no cuenta con que alguien va a construir dos listas seguidas independientes, y el contador de los elementos es inconsciente del número concreto que pongamos. Dicho de otra forma, cuando Markdown ve un número, el que sea, lo toma por "esto es un elemento de una lista numerada", pero la numeración, el conteo, lo hace automáticamente, sin consideración del número particular que se ponga. Pandoc, supuestamente, proporciona una extensión que tiene en cuenta el número concreto que indiquemos y hay varias formas de resolver este problema.

El segundo problema tiene que ver con el espaciado vertical. Ciertamente lo que son distintas secciones del documento: título, orden del día, exposición, cierre de sesión y firma, no han sido etiquetadas como tales secciones y el texto de unas se agolpa en el de las otras.

Podemos crear secciones con títulos invisibles con las mismas marcas de secciones que conocemos, de manera tal que nuestro documento siguiese este esquema:

```
# Cabecera principal
```

Gómez Varas Patricio
Rodríguez García Julia Beatríz
Sanjuán Pernas Luis

ACTA DE LA REUNIÓN DEL DEPARTAMENTO DE GUITARRA CELEBRADA EL DIA 30 DE SEPTIEMBRE DE 2014

Se inicia la sesión a las 20:00 horas del 30 de septiembre de 2014 en el aula 18 del centro.

ORDEN DEL DÍA

1. Lectura y aprobación, si procede, del acta de la reunión anterior.
2. Comentario del borrador de la programación.
3. Otros asuntos
4. Lectura y aprobación, si procede, del acta de la reunión anterior.

   Se lee el acta de la reunión anterior, que se aprueba.
5. Comentario del borrador de la programación.

   Se pone en conocimiento de Patricio del borrador de la nueva programación. No hay ningún cambio esencial con respecto de la del curso pasado. Pero hay muchos cambios formales, particularmente en la programación de conjunto.

   Asimismo, se decide (como quedó comentado en la última reunión del curso pasado) mantener la idea de una audición pública final donde el grueso de la participación corresponderá a grupos, surgidos de las clases colectivas y la de conjunto.
6. Otros asuntos.

   - Se decide fecha para la audición final, que será, dependiendo de la ocupación, en el aula de Orquesta o de Coro, un viernes en torno a las 18.00h (para facilitar la participación de los más pequeños) y a principios de junio. Se trasmitirá a Jefatura dicha solicitud.
   - En otro orden de cosas, se informa de la novedad para este curso de que la memoria del departamento debe incluir un adjunto con las faltas de los alumnos durante el curso.

Y sin más asuntos que tratar se cierra la sesión a las 21:00 horas del 30 de septiembre de 2014.
El Jefe del Departamento:
Luis Sanjuán

1

Figura 2: acta_v1.pdf

```
## Orden del día

Aquí va el orden del día

##

Aquí va la exposición

##

Aquí va el cierre de sesión

##

Aquí iría la firma
```

Notad que uso las marcas de sección, pero sin titular las secciones para atenerme al modelo, donde tales secciones vienen sin títulos.

Hago las modificaciones correspondientes en nuestro fichero **acta_v1.md**, lo guardo como **acta_v2.md** y vuelvo a ejecutar la instrucción anterior `pandoc` con el fichero de salida como **acta_v2.pdf** y el fichero de entrada **acta_v2.md**. Se obtiene el resultado que se muestra en la página siguiente.

El problema del espaciado ha desaparecido y, como por arte de magia, también el problema en la numeración de las dos listas. Esto segundo es así porque sucesiones de elementos de lista pertenecen a distintas listas si dichas sucesiones forman parte de distintas estructuras, en este caso subsecciones. Así, por tanto, hemos resuelto dos problemas en uno, por el simple hecho de estructurar nuestros documentos, en lugar de escribir sin atender a la lógica interna de lo que escribimos.

Un par de retoques (con LaTeX) finalizan la emulación del modelo. El primero es la línea horizontal que en el modelo separa el orden del día de la exposición. Aunque creo que esto fue un añadido mío y no forma parte del modelo que inicialmente me entregó Alberto. El segundo retoque es añadir más espacio vertical para la firma, particularmente para la mía que ocupa bastante.

Sabemos que podemos añadir instrucciones LaTeX dentro de documentos Markdown y que van a funcionar. Lo hemos comentado de pasada en el foro.

Los fragmentos pertinentes de nuestro documento con las instrucciones LaTeX incorporadas son los siguientes.

Para añadir la línea de separación tras el orden del día:

```
## Orden del día

1.  Lectura y aprobación, si procede, del acta de la reunión anterior.
2.  Comentario del borrador de la programación.
3.  Otros asuntos

\begin{flushright}\rule{5mm}{.5mm}\end{flushright}
```

Gómez Varas
Patricio
Rodríguez García
Julia Beatríz
Sanjuán Pernas
Luis

ACTA DE LA REUNIÓN DEL DEPARTAMENTO DE GUITARRA CELEBRADA EL DIA 30 DE SEPTIEMBRE DE 2014

Se inicia la sesión a las 20:00 horas del 30 de septiembre de 2014 en el aula 18 del centro.

ORDEN DEL DÍA

1. Lectura y aprobación, si procede, del acta de la reunión anterior.
2. Comentario del borrador de la programación.
3. Otros asuntos

1. Lectura y aprobación, si procede, del acta de la reunión anterior.

   Se lee el acta de la reunión anterior, que se aprueba.

2. Comentario del borrador de la programación.

   Se pone en conocimiento de Patricio del borrador de la nueva programación. No hay ningún cambio esencial con respecto de la del curso pasado. Pero hay muchos cambios formales, particularmente en la programación de conjunto.

   Asimismo, se decide (como quedó comentado en la última reunión del curso pasado) mantener la idea de una audición pública final donde el grueso de la participación corresponderá a grupos, surgidos de las clases colectivas y la de conjunto.

3. Otros asuntos.

   - Se decide fecha para la audición final, que será, dependiendo de la ocupación, en el aula de Orquesta o de Coro, un viernes en torno a las 18.00h (para facilitar la participación de los más pequeños) y a principios de junio. Se trasmitirá a Jefatura dicha solicitud.
   - En otro orden de cosas, se informa de la novedad para este curso de que la memoria del departamento debe incluir un adjunto con las faltas de los alumnos durante el curso.

Y sin más asuntos que tratar se cierra la sesión a las 21:00 horas del 30 de septiembre de 2014.

El Jefe del Departamento:
Luis Sanjuán

Figura 3: acta_v2.pdf

Para añadir espacio vertical para la firma:

```
El Jefe de Departamento:

\vspace*{3cm}

Luis Sanjuán
```

No me detengo en esto, pues son cosas específicas de LaTeX.

Si proceso de nuevo el documento con `pandoc` con estas modificaciones y cambiando los nombres de archivos como corresponda, obtengo el resultado final, que se muestra en las dos páginas siguientes. Ocupa dos páginas esta vez. No está mal. Así comprobamos también que la plantilla sigue funcionando con documentos multi-página.

8.5. El logo. Marcas de agua con PDFtk

Para incluir el logotipo del centro en la parte superior del acta disponemos, como comentaba al principio, de varias opciones. Podríamos usar el paquete `background` u otros paquetes especializados de LaTeX como el paquete `watermark`. Pero no necesitamos un control tan fino. Bastará con una herramienta de propósito general para la manipulación de **pdf**s, como es PDFtk, que además es multiplataforma. Más información sobre PDFtk en su página web:

https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/

PDFtk es muy útil para muchas cosas que tienen que ver con manipulación de ficheros **pdf**. Para incluir marcas de agua desde la línea de comandos, la instrucción es la siguiente:

```
pdftk <fichero-input>.pdf background <marca-agua>.pdf output <fichero-output>.pdf
```

He creado un fichero **pdf** a partir del logotipo con el nombre de fichero **membrete.pdf**, que está en el mismo directorio que el resto de ficheros relativos a actas que hemos visto. A partir de nuestro último **pdf** que, recuerdo, se llamaba **acta_v3.pdf** voy a crear un fichero al que añadiré el logotipo y que llamaré **acta_v4.pdf**. La instrucción será la siguiente:

```
pdftk acta_v3.pdf background membrete.pdf output acta_v4.pdf
```

El resultado de la primera página (la segunda página también lo incluye) aparece en la figura 6.

Es suficiente ver que funciona como esperamos. Detalles no significativos para el propósito de la exposición como un encuadre mejor acabado entre el logotipo y la página, la dimensiones exactas de la página y de sus elementos, el grosor de la línea de separación, etc. son cuestiones que se pueden modificar ya sea ajustando la imagen del logotipo, ya refinando las dimensiones de la página. Se trata de jugar con números hasta obtener lo que se ajusta a nuestros gustos.

Gómez Varas Patricio
Rodríguez García Julia Beatríz
Sanjuán Pernas Luis

ACTA DE LA REUNIÓN DEL DEPARTAMENTO DE GUITARRA CELEBRADA EL DIA 30 DE SEPTIEMBRE DE 2014

Se inicia la sesión a las 20:00 horas del 30 de septiembre de 2014 en el aula 18 del centro.

ORDEN DEL DÍA

1. Lectura y aprobación, si procede, del acta de la reunión anterior.
2. Comentario del borrador de la programación.
3. Otros asuntos

—

1. Lectura y aprobación, si procede, del acta de la reunión anterior.

   Se lee el acta de la reunión anterior, que se aprueba.

2. Comentario del borrador de la programación.

   Se pone en conocimiento de Patricio del borrador de la nueva programación. No hay ningún cambio esencial con respecto de la del curso pasado. Pero hay muchos cambios formales, particularmente en la programación de conjunto.

   Asimismo, se decide (como quedó comentado en la última reunión del curso pasado) mantener la idea de una audición pública final donde el grueso de la participación corresponderá a grupos, surgidos de las clases colectivas y la de conjunto.

3. Otros asuntos.

   - Se decide fecha para la audición final, que será, dependiendo de la ocupación, en el aula de Orquesta o de Coro, un viernes en torno a las 18.00h (para facilitar la participación de los más pequeños) y a principios de junio. Se trasmitirá a Jefatura dicha solicitud.
   - En otro orden de cosas, se informa de la novedad para este curso de que la memoria del departamento debe incluír un adjunto con las faltas de los alumnos durante el curso.

Y sin más asuntos que tratar se cierra la sesión a las 21:00 horas del 30 de septiembre de 2014.

Figura 4: acta_v3.pdf - pag. 1

El Jefe del Departamento:

Luis Sanjuán

Figura 5: acta_v3.pdf - pag. 2

Junta de Castilla y León

Gómez Varas Patricio
Rodríguez García Julia Beatríz
Sanjuán Pernas Luis

ACTA DE LA REUNIÓN DEL DEPARTAMENTO DE GUITARRA CELEBRADA EL DIA 30 DE SEPTIEMBRE DE 2014

Se inicia la sesión a las 20:00 horas del 30 de septiembre de 2014 en el aula 18 del centro.

ORDEN DEL DÍA

1. Lectura y aprobación, si procede, del acta de la reunión anterior.
2. Comentario del borrador de la programación.
3. Otros asuntos

—

1. Lectura y aprobación, si procede, del acta de la reunión anterior.

   Se lee el acta de la reunión anterior, que se aprueba.

2. Comentario del borrador de la programación.

   Se pone en conocimiento de Patricio del borrador de la nueva programación. No hay ningún cambio esencial con respecto de la del curso pasado. Pero hay muchos cambios formales, particularmente en la programación de conjunto.

   Asimismo, se decide (como quedó comentado en la última reunión del curso pasado) mantener la idea de una audición pública final donde el grueso de la participación corresponderá a grupos, surgidos de las clases colectivas y la de conjunto.

3. Otros asuntos.

   - Se decide fecha para la audición final, que será, dependiendo de la ocupación, en el aula de Orquesta o de Coro, un viernes en torno a las 18.00h (para facilitar la participación de los más pequeños) y a principios de junio. Se trasmitirá a Jefatura dicha solicitud.
   - En otro orden de cosas, se informa de la novedad para este curso de que la memoria del departamento debe incluir un adjunto con las faltas de los alumnos durante el curso.

Y sin más asuntos que tratar se cierra la sesión a las 21:00 horas del 30 de septiembre de 2014.

1

Figura 6: acta_v4.pdf - pag. 1

8.6. Variables

Hasta aquí el trabajo ha sido completado con éxito. Pero al sufrido jefe de departamento le surgirá una pregunta ineludible. Bien, ahora ya puedo escribir las actas mediante Markdown y quitarme un montón de problemas con la edición directa de **pdf**s. Pero ya que estamos, ¿no sería genial evitar también el trabajo repetitivo de incluir los profesores, ya sea volviendo a escribir sus nombres o borrando a los no asistentes? ¿Y qué con todas esas fechas y horas repetidas una y otra vez a lo largo del documento?

Pues sí, se puede, y para eso están las variables de Pandoc y los bloques de metadatos. Hasta ahora, en una práctica anterior, hemos aprendido a usar variables preestablecidas por Pandoc. Pero Pandoc, a través de sus plantillas, permite también crear nuestras propias variables e incluir sus valores en ficheros de metadatos.

En la práctica anterior sobre este tema, para simplificar, comenté que el bloque de metadatos debería estar al principio del documento que se procesa. Recordamos que había una marca para ello:

```
---
Aquí van los campos de metadatos
---
```

Pero también es posible crear un fichero independiente de metadatos.

Tampoco me voy a detener en este punto, salvo que alguno de vosotros me indique que quiere profundizar en él. El fichero de metadatos tiene la extensión **yaml**. Voy a llamar a nuestro fichero de metadatos para el acta **variables.yaml** y su contenido es el siguiente:

```
---
day: 30
month: Septiembre
year: 2014
start: 20:00
end: 21:00
prof:
- a: Gómez Varas
  n: Patricio
- a: Rodríguez García
  n: Julia Beatríz
- a: Sanjuán Pernas
  n: Luis
---
```

Cada campo tiene un nombre. He escogido un nombre en inglés para que sea más fácil de diferenciar en la plantilla, pero podría ser uno en español, aunque sin acentos ni la ñ. Tras cada campo separado por dos puntos va su valor. Cuando los campos contienen varios valores estos se se marcan con guión. Se permite

también que los campos tengan miembros. Cada miembro de un campo tiene a su vez su nombre, aquí `a` por apellido y `n` por nombre. Este formato no es Markdown, sino YAML:

http://www.yaml.org/

Se usa en varios ámbitos, y Pandoc lo soporta para proporcionar su opción de metadatos y variables en plantillas.

Para que el tinglado funcione hay que hacer dos cosas más:

- Sustituir los profesores por variables en la plantilla LaTeX.
- Incluir nuevas zonas con variables en la plantilla, precisamente aquellas que no cambiar de una acta a otra.

Respecto de este último punto. Si pensamos en sustituir los valores de fechas y horas por variables, resulta que son zonas del documento que no cambiarán de un acta a otra y que, por tanto, son perfectamente susceptibles de ser incluidas en la plantilla.

Nuestra plantilla se debe transformar del modo siguiente para ganar la flexibilidad perseguida:

```
\begin{document}
\marginnote{\small \mbox{}$for(prof)$$prof.a$\\$prof.n$\\ $endfor$}

\section{Acta de la reunión del departamento de Guitarra del $day$ de $month$ de $year$ ..

Se inicia la sesión a las $start$ horas ...

$body$

\subsection{}

Y sin más asuntos que tratar se cierra la sesión a las $end$ horas ...

\subsection{}

El Jefe del Departamento

\vspace*{3cm}

Luis Sanjuán
\end{document}
```

Por su parte, el acta misma, nuestro documento, que ahora llamaré **`acta.md`** queda simplificada notablemente en su versión definitiva, pues partes de ella han ido a parar a la plantilla misma:

```
## Orden del día

1.  Lectura y aprobación, si procede, del acta de la reunión anterior.
2.  Comentario del borrador de la programación.
3.  Otros asuntos

\begin{flushright}\rule{5mm}{.5mm}\end{flushright}

##

1.  Lectura y aprobación, si procede, del acta de la reunión anterior.

    Se lee el acta de la reunión anterior, que se aprueba.

2.  Comentario del borrador de la programación.

    Se pone en conocimiento de Patricio del borrador de la nueva programación.
    No hay ningún cambio esencial con respecto de la del curso pasado.
    Pero hay muchos cambios formales, particularmente en la programación
    de conjunto.

    Asimismo, se decide (como quedó comentado en la última reunión del
    curso pasado) mantener la idea de una audición pública final donde el
    grueso de la participación corresponderá a grupos, surgidos de las
    clases colectivas y la de conjunto.

3.  Otros asuntos.

    - Se decide fecha para la audición final, que será, dependiendo de
      la ocupación, en el aula de Orquesta o de Coro, un viernes en torno
      a las 18.00h (para facilitar la participación de los más pequeños) y a
      principios de junio. Se trasmitirá a Jefatura dicha solicitud.

    - En otro orden de cosas, se informa de la novedad para este curso de
      que la memoria del departamento debe incluir un adjunto con las faltas
      de los alumnos durante el curso.
```

Este último tipo de documento es lo que tendría que escribirse cada vez que se redacta un acta. El resto está automatizado gracias a Pandoc y a nuestra plantilla.

Notad bien dos cambios sustanciales en la plantilla, aparte de las variables y secciones añadidas.

- La variable `$body$` va exactamente allí donde irá el contenido sin variable alguna del acta, o sea, lo que redactará el jefe de departamento.
- Las etiquetas de sección y subsección de Markdown, `#` y `##`, respectivamente han sido sustituidas donde corresponde por sus equivalentes en LaTeX `\section` y `\subsection`. Esto último es compresible, puesto que

nuestra plantilla es una plantilla LaTeX, que debe escribirse enteramente con marcas de LaTeX.

Recordemos los documentos que hemos generado y los ficheros con los que estamos trabajando en su forma final:

- **`acta.md`**: el acta final en cuanto tal, escrita en Markdown
- **`plantilla_acta.latex`**: la plantilla en LaTeX
- **`variables.yaml`**: el bloque de metadatos
- **`membrete.pdf`**: el **`pdf`** que contiene el logo para la marca de agua

La instrucción `pandoc` para generar el acta final es ligeramente diferente a las que hemos visto, pues debe añadir al final el nombre del fichero del bloque de metadatos. Sería, en definitiva, la siguiente:

```
pandoc -s -o acta.pdf --template plantilla_acta.latex acta.md variables.yaml
```

El resultado **`acta.pdf`** es exactamente igual que el de las figuras 4 y 5 anteriores, es decir el acta sin la marca de agua. Sobre este **`pdf`** es sobre el que habría que incluir la marca de agua como antes describimos, mediante la instrucción:

```
pdftk acta.pdf background membrete.pdf output acta_con_logo.pdf
```

Cambié el orden de la exposición natural e introduje antes el asunto de la marca de agua para dejar este aspecto de las variables, más difícil, para el final.

8.7. Aplicaciones prácticas

Éste u otro tipo de exposiciones similares sobre otro tipo de documentos muestran de qué forma sería posible crear un proceso para la generación de documentos oficiales de manera que el trabajo no fuera oneroso para nadie y estuviera distribuido.

1. Alguien con suficientes conocimientos y tiempo crearía la infraestructura, en este caso, construiría la plantilla de LaTeX para Pandoc y el esqueleto del fichero que contiene el campo de metadatos.
2. El profesor redactaría de una forma muy simple su documento, en este caso, escribiría el acta y rellenaría el campo de metadatos. En caso de documentos conjuntos, los distintos profesores modificarían sin dificultad un texto plano.
3. El responsable de producir los documentos finales se limitaría a ejecutar `pandoc` y `pdftk` y a imprimir los **`pdf`**s.

En este ciclo la parte más trabajosa y que lleva más tiempo es la creación de la plantilla. Pero también es cierto que, una vez creadas, es trabajo que revierte en el futuro reduciendo su lado repetitivo y engorroso.

En el foro adjunto los ficheros finales arriba comentados por si alguno quiere experimentar. Si el procesamiento falla puede ser porque en vuestra instalación básica de TEX falte alguno de los paquete LATEX que he aplicado. En tal caso, sería simplemente cuestión de instalarlos.

9. Apéndice: Automatización

No me quedaba a gusto sin comentar un aspecto más, también avanzado, pero que creo puede ser interesante de cara a disponer de argumentos en defensa de estas herramientas, o, incluso, puede seros útil si algún día intentáis entrar a fondo en estos derroteros por vuestra cuenta. Una utilidad inmediata puede haberla particularmente para Enrique, ya que es jefe de departamento, y, quizá, en su sistema operativo, lo que voy a describir también puede funcionar.

9.1. Introducción

Una brevísima introducción sobre la filosofía de Unix. Antes que nada comentar que llamo Unix a los programas que se basan en las ideas y código original de quienes desarrollaron Unix, allá por los 70 del siglo pasado, los mismos que cocrearon el lenguaje de programación C, que sigue siendo, según las estadísticas, en el que más se escribe hoy en día. No conviene tampoco olvidar que la Red global tal como la conocemos debe mucho a esta época y a esta filosofía.

Resulta que estos lumbreras, Dennis Ritchie y Ken Thompson, que trabajaban entonces en los laboratorios Bell, idearon algo realmente bueno, pues ha mostrado su excelencia y resistencia al paso del tiempo. Hoy en día los herederos popularmente más conocidos de aquel Unix original son los entornos Linux y MacOSX. Aquí habría que hacer muchas matizaciones. Quedémonos con que aquella filosofía no sólo persiste, sino que muestra día a día su excelencia.

Entre el ideario Unix están estos principios que tienen especial relevancia para el asunto que nos toca:

1. El texto plano es el medio central en que se almacenan los datos que los programas manipularán e intercambiarán.
2. Las herramientas que manipulan ese texto son pequeños programas que hacen una sola cosa, pero que la hacen bien.

De aquí se sigue que la complejidad de un proceso se reduce a la colaboración de pequeños programas especializados que trabajan sobre texto plano, texto plano cuyo formato debe ser simple y fácil de analizar y manipular programáticamente.

En concreto, cada uno de estos pequeños programas recibe un input en texto plano fácilmente procesable y ejecuta sobre él la operación para la que está

diseñado. El resultado es otro texto plano que será consumido por otro programa pequeño con otra función específica. La colaboración se prolonga hasta obtener el output deseado.

¿Por qué esto es relevante para nosotros? Por dos cosas.

Primero, porque, como vamos viendo, nos estamos aprovechando de esta filosofía. Editamos un texto plano en un formato simple y fácil de analizar (Markdown). El resultado se lo pasamos a `pandoc`. `pandoc` analiza esas marcas y genera otro texto plano que, en el caso de conversión a `pdf`, se lo pasa a la máquina LaTeX. LaTeX, por su parte, lo convierte en formato TeX para que una máquina TeX lo procese. Finalmente, en nuestra anterior práctica, el `pdf` resultante es manipulado por otro programa, PDFtk, para insertarle la marca de agua.

Cada programa realiza su función específica, y todos pueden colaborar porque conocen las convenciones del texto que cada uno maneja, convenciones que, por norma, tratan de ser lo más simple posibles, con el fin de que ese texto sea manipulado con facilidad.

La segunda razón es que este tipo de filosofía permite formas relativamente cómodas o asequibles de automatización. Puesto que todo son textos fácilmente analizables, siempre podemos encontrar mecanismos para automatizar operaciones repetitivas sobre ellos, mecanismos que no impliquen crear grandes y complejos programas cuya implementación exija un esfuerzo humano (de número de personas y de horas de trabajo) demasiado grande para ser practicable.

9.2. Automatizar la generación de actas

Como ejemplo de ello os muestro cómo tengo automatizada la generación de las actas. Lo que vale para las actas sería aplicable, *mutatis mutandi*, a otras situaciones parecidas.

En realidad, mi automatización era otra en el origen. No utilizaba Pandoc, sino LaTeX puro. Ha sido a propósito de las prácticas en el seno del grupo que me he planteado confiar a Pandoc el grueso de la tarea.

El problema inicial que me planteaba, aparte de lo comentado ya en el texto del otro día, era el siguiente. Está bien producir un pdf de la forma que hemos explicado, pero todavía hay cosas tediosas que podrían automatizarse. En particular:

1. Los puntos del orden del día se repiten tal cual en el cuerpo del acta. Lo ideal sería escribir sólo ese cuerpo, sin necesidad de cortar y pegar. Lo ideal sería que un programa generase la lista que contiene el orden del día y que esa lista se insertase antes del cuerpo del acta. El jefe de departamento se limitaría a escribir el contenido del acta, nada más. Otra ventaja de ello es que no tendría por qué haber ninguna marca extraña al puro Markdown dentro del acta, pues todas esas marcas e instrucciones de LaTeX quedarían desplazadas a la plantilla de LaTeX. Las actas serían Markdown puro.
2. A veces tengo varias actas redactadas. Es aburrido tener que generar un pdf para cada una de las actas redactadas. Lo ideal sería que todos los pdfs de todas las actas se generasen en una sola operación automáticamente.

Tratar de resolver estos problemas sería difilísimo en un entorno que no fuera el entorno reducido de textos planos que siguen convenciones sencillas y simples. Por ejemplo, los procesadores de textos utilizan marcas XML, pero la cantidad y especificaciones de esas marcas es tal que, en esos entornos, no resulta factible plantearse una solución que pueda diseñarse en un tiempo razonable.

9.2.1. Generación automática del orden del día

El primero de los problemas se puede expresar más específicamente en la filosofía anteriormente descrita como una tarea que realizar:

> Convertir un input dado, que contiene el cuerpo del acta, en un output, que contenga sólo el orden del día. Posteriormente, incluir ese orden del día de una forma automática.

Lo primero es fácil de conseguir mediante un filtro de Unix, que toma el cuerpo del acta (escrito en Markdown) y selecciona aquellas líneas en él que tienen un dígito como primer carácter de la línea, seguido de punto, seguido de uno o más espacios y seguido de cualquier sucesión de caracteres alfanuméricos. Esta descripción corresponde precisamente a lo que en Markdown es un ítem de lista numerada de primer nivel, justo la forma en que hay que etiquetar, siguiendo la convención del Conservatorio, los ítems de la lista del orden del día y los ítems del cuerpo del acta a los que se añadirá su correspondiente comentario.

Por si interesa, el filtro es éste (funciona, aunque sea críptico):

```
grep '^[[:digit:]]\. \+[[:alpha:]]' acta > acta_orden_del_dia
```

9.2.2. Generación de un bloque de metadatos para el acta

El segundo problema principal tiene que ver con extraer información de la fecha y hora para cada acta.

Para almacenar dicha información se debe pensar en alguna convención sencilla de forma que, dada un acta, un pequeño programa pudiera extraer la información necesaria concerniente a su fecha y hora.

Hay varias formas de implementar una convención así. Una natural sería crear una especie de mini base de datos. Algo como esto:

```
#acta;dia;mes;año;hora comienzo;hora fin
acta1;12;septiembre;2014;12:00;13:00
acta2:13;octubre;2014;13:00;14:00
...
```

De esta mini base de datos se podría obtener dicha información.

Lo primero que se me ocurrió, sin embargo, fue algo más críptico, pero a la vez más breve: codificar la información relevante a cada acta en su nombre de fichero. Por ejemplo, el acta primera tendría un nombre de fichero como el siguiente:

```
a01_1209141200.md
```

que significa: acta1, del día 12 del mes 09 del año 14, comienzo 12:00h. Por cierto, en mi implementación he suprimido la fecha de cierre de sesión y la computo como una 1h más que la de inicio, que viene a ser aproximadamente eso. Pero sería una tarea pendiente implementar una fecha de cierre exacta.

Suponiendo que hemos extraído adecuadamente esa información, ¿qué hacer con ella? Si recordamos lo del día pasado, esa información debía introducirse como los valores respectivos en el bloque de metadatos **`variables.yaml`**.

En definitiva, la solución del problema se puede describir como una composición de dos tareas:

> (Primero): Extraer del nombre de fichero la información relevante y convertirla de modo adecuado. Por ejemplo los dígitos '09' correspondientes al mes, deben extraerse y convertirse a 'septiembre'.
>
> (Segundo): Introducir los datos extraídos y adecuadamente convertidos en los correspondientes campos del bloque de metadatos perteneciente al acta del caso.

De nuevo, las dos tareas son viables porque se basan en secuencias de texto plano que obedecen convenciones simples. En el primer caso, se trata de obtener datos de un formato plano basado en una convención simple:

```
nombreacta_dia(dos dígitos)mes(dos dígitos)año(dos dígitos)hora(cuatro dígitos)
```

En el segundo, de insertar datos en un texto que a su vez sigue una convención simple:

```
nombre_del_campo: valor_del_campo
```

En otras palabras, gracias a la simplicidad de la convención, es sencillo generar un fichero **`yaml`** donde los datos procedentes del nombre de fichero del acta acaben insertados como sigue:

```yaml
# fichero a1.yaml

day: 12
month: septiembre
year: 2014
start: 12:00
end: 13:00
```

9.2.3. Variable para el fichero que contiene el orden del día

Dando por hecho que hemos conseguido crear un fichero `md` que contiene el orden del día a partir del fichero que contiene el cuerpo del acta, podemos recurrir de nuevo a las variables y plantillas LaTeX de Pandoc con la idea de introducir una nueva variable en la plantilla, correspondiente al nombre del fichero que contiene el acta. Esta variable se declararía, a su vez, en el bloque de metadatos del acta del caso de un modo parecido al siguiente:

```
# fichero a01_1209141200.yaml

day: 12
...
od: <nombre_del_fichero_que_contiene_el_orden_del_dia>
...
```

Por su parte, en la plantilla podríamos incluir el contenido de ese fichero con una instrucción `\input` de LaTeX. Esta sería la parte de la plantilla relevante con dicho añadido:

```
% plantilla_acta.latex
...
\subsection{Orden del día}
\input{$od$}

\begin{flushright}\rule{5mm}{.5mm}\end{flushright}
...
```

La segunda línea incluye el fichero del orden del día, a través de la variable `$od$`, que acabamos de crear en el bloque de metadatos del acta.

He añadido de paso la siguiente línea de la plantilla con la intención de destacar una ventaja más, una esencial, que hemos ganado. Si recordáis, la línea final de este fragmento era la que producía la pequeña línea de separación entre el orden del día y el cuerpo del acta. Al ser capaces, en esta versión de la plantilla, de introducir programáticamente el orden del día sin intervención del usuario, ya no queda rastro de nada que no sea puro Markdown en el contenido del acta. Dicho de otra forma, lo que realmente va a redactar el jefe de departamento es el cuerpo del acta en Markdown puro:

```
1.  Lectura y aprobación, si procede, del acta de la reunión anterior.

    Se lee el acta de la reunión anterior, que se aprueba.

2.  Comentario del borrador de la programación.

    Se pone en conocimiento de Patricio del borrador de la nueva programación.
    No hay ningún cambio esencial con respecto de la del curso pasado.
    Pero hay muchos cambios formales, particularmente en la programación
```

```
    de conjunto.

    Asimismo, se decide (como quedó comentado en la última reunión del
    curso pasado) mantener la idea de una audición pública final donde el
    grueso de la participación corresponderá a grupos, surgidos de las
    clases colectivas y la de conjunto.

3.  Otros asuntos.

    - Se decide fecha para la audición final, que será, dependiendo de
      la ocupación, en el aula de Orquesta o de Coro, un viernes en torno
      a las 18.00h (para facilitar la participación de los más pequeños) y a
      principios de junio. Se trasmitirá a Jefatura dicha solicitud.

    - En otro orden de cosas, se informa de la novedad para este curso de
      que la memoria del departamento debe incluir un adjunto con las faltas
      de los alumnos durante el curso.
```

El jefe del departamento sólo tendría una o dos tareas más:

1. Nombrar los ficheros de sus actas siguiendo la convención referida antes.
2. En caso de que tuviera ausencias de profesores a la reunión, editar, antes de ejecutar el programa, el fichero **`variables.yaml`** *comentando* aquellos profesores de su departamento que no asistieron. *Comentar* es una palabra técnica que se refiere a poner un carácter especial delante de una línea. Ese carácter que, dependiendo del formato del fichero o lenguaje en que está escrito, es uno u otro, hace que la línea precedida por él no exista para las programas que lo procesan. En el caso de un fichero **`yaml`**, el carácter para comentar una línea es `#`. Así, por ejemplo, en mi fichero **`variables.yaml`**, si tuviese que ocultar a Patricio, porque no hubiese asistido, para que no apareciese en la nota al margen de los asistentes, tendría que hacer esto:

```
# fichero variables.yaml
...
prof:
#- a: Gómez Varas
#  n: Patricio
- a: Rodríguez García
  n: Julia Beatríz
- a: Sanjuán Pernas
  n: Luis
```

Idealmente, mi programa debería ser flexible y aceptar opciones, como hace `pandoc`, de forma que pudiese decir, por ejemplo:

```
generar_acta --excluir Patricio a01_1209141200.md
```

Quede esto como tarea pendiente para el futuro.

9.2.4. Por qué a veces es conveniente no usar `--standalone` con `pandoc`

Pensemos de nuevo en el tipo de documento que es nuestra **`plantilla_acta.latex`**. Como su extensión claramente indica, es un documento de LaTeX. Si alguno ha seguido hasta aquí la explicación, le habrá surgido la siguiente duda:

> Antes hicimos que se crease programáticamente un fichero que contiene el orden del día. Ese fichero resultaba de filtrar las líneas adecuadas del fichero del acta, que es un fichero `markdown`. Puesto que el fichero de origen es `markdown`, lo que resulta de aplicarle el filtro será necesariamente un fichero `markdown`. Después, hemos incluido el contenido de ese fichero en la plantilla con la instrucción `\input` de LaTeX. Aquí hay algo que no cuadra. ¿No estamos diciendo que el fichero LaTeXdebe contener etiquetas LaTeXúnicamente y no etiquetas Markdown?

Si alguno de vosotros se ha hecho esta reflexión, enhorabuena, ha sido muy agudo y da plenamente en el clavo.

Evidentemente, no podemos incluir Markdown directamente en LaTeX. Ya vimos el otro día, por ejemplo, que las etiquetas Markdown para encabezados debían, al pasarse a la plantilla, etiquetarse con sus correspondientes equivalentes LaTeX (`\section`, `\subsection`, etc). Con el fichero **`md`** del orden del día sucede lo mismo. La lista y sus elementos deben etiquetarse con sus equivalentes para LaTeX si queremos que todo funcione correctamente en nuestra plantilla LaTeX. En consecuencia, no podemos incluir allí directamente nuestro orden del día en formato Markdown, tenemos que convertir antes ese formato a LaTeX. ¿Cómo hacerlo? ¿A mano? Esto no parece tan simple como cambiar `'#'` por `'\section'`. De hecho sería un engorro hacerlo a mano. Lo podemos hacer programáticamente por nuestra cuenta, pero eso también es absurdo, porque `pandoc` puede hacerlo por nosotros:

```
pandoc -o orden_del_dia.tex orden_del_dia.md
```

¿En qué es diferente este comando respecto de los que ya conocemos? En dos cosas:

- La extensión del fichero de salida es **`tex`**, que quiere decir documento de LaTeX. Estamos, pues, convirtiendo de `markdown` a `latex`, algo que todavía no habíamos hecho, puesto que no lo habíamos necesitado.
- No estamos aplicando la opción `--standalone`, o `-s` en su versión abreviada. Y no lo hacemos, porque no queremos que `pandoc` genere un preámbulo para el resultado. Necesitamos únicamente la lista sin preámbulo, puesto que esa lista (con etiquetas LaTeX) la vamos a incluir en un fichero que consta ya de su propio preámbulo, nuestra plantilla LaTeX, y ningún documento LaTeXcorrecto puede tener más que un único preámbulo.

En general, no usar `--standalone` con `pandoc` es apropiado cuando queremos obtener algo que será insertado en un documento más grande, este sí, independiente. En el caso que analizamos, es un documento LaTeX; en otros casos podría ser, por ejemplo, HTML para ser incluido en una página web más grande.

9.2.5. Procesar todas las actas de un golpe

Una vez que hemos creado un script siguiendo la lógica descrita, llamémosle `generar_acta.sh`, que permite producir un `pdf` del acta de la forma automatizada que hemos previsto, conseguir que todas las actas se generen a la vez es fácil en entornos Unix. Estos entornos proporcionan constructos de línea de comandos, llamados *bucles*, para este tipo de tareas. En el nuestro, se podría hacer de la siguiente forma, asumiendo que estamos en el directorio que contiene todas las actas en formato `md`:

```
for acta in $(ls *.md); do ./generar_acta.sh $acta; done
```

El pero es que esta instrucción asume que todos los profesores han asistido a las reuniones correspondientes. Para que fuera sensible a las ausencias, es necesario flexibilizar y mejorar el script. Tarea pendiente.

9.3. Epílogo

Todo esto parece endemoniadamente complicado. Os aseguro que no lo es. Los comandos necesarios para ejecutar todas estas tareas son comandos que todo usuario intermedio de Linux que trabaje habitualmente y predominantemente en el terminal conoce y sabe utilizar. Lo que lleva más trabajo es organizarlo todo en un script que sea fácil de entender, modificar y extender en el futuro.

Lo que me importa destacar es que las posibilidades de automatización se abren porque los programas que empleamos y los formatos y convenciones de esos formatos son fieles a las ideas brillantes que pergeñaron los creadores de Unix y del lenguaje C.

No estamos ante programas mastodónticos (un procesador de texto lo es: millones de líneas de código), sino programas pequeños y especializados que se combinan como piezas de un Lego. Incluso la instalación completa TEX{}Live, que pesa tanto en megas, es tan grande porque recoge una infinidad de pequeños paquetes que expanden lo que el ofrece núcleo de TEX / LATEX, muy pequeño en sí mismo.

9.4. Aplicación práctica

El script, unido a los ficheros que comenté el otro día, son aplicables directamente en cualquier plataforma (Linux, quizá también MacOSX) que tenga instalado Pandoc, PDFtk, TEX, los paquetes de TEX mencionados el otro día, `bash` y algunos mini-programas, típicos de Unix, que utilizo en el script.

La plantilla está retocada respecto de la que presente en la sección anterior. Ahora tiene en cuenta la geometría del modelo original, utiliza fuentes más parecidas a las del modelo, y produce un resultado tan semejante que no es fácil decidir cuál es el modelo y cuál el original.

Como referencia, adjuntaré en el foro un archivo comprimido con todos los ficheros necesarios, entre los cuales se incluye una explicación de su contenido y uso.

10. Apéndice: condicionales y bucles en Pandoc

En este apéndice comento los típicos constructos que aparecen en las plantillas de Pandoc. La documentación oficial es concisa y no propone apenas ejemplos. Supone un lector que ya entiende con claridad la función de estos constructos y que es capaz de realizar experimentos por sí mismo para confirmar su significado y aplicación. Salvo programadores, es difícil que usuarios avanzados con interés sean capaces de obtener una guía suficiente de dicha documentación. Este apéndice propone ejemplos de uso para intentar hacer poco más digerible estas características avanzadas de Pandoc. Aspectos relativos a LaTeX no se comentan.

Nota: Cuando los comandos son muy largos los divido a través de un \, como es convencional en Unix.

10.1. Variables en Pandoc

Una variable en Pandoc tiene la siguiente sintaxis:

```
$nombre-de-variable$
```

Cuando ejecutamos `pandoc` cada variable de la plantilla se substituye por su valor. Este valor se puede pasar a `pandoc` de distintas formas. Una de ellas, como veremos más adelante, es pasárselo a través de la opción de línea de órdenes `-M nombre-de-variable=valor`(donde `-M` es la forma abreviada de `--metadata`).

Por lo respecta a las variables pre-definidas por Pandoc y que están incluidas en la plantilla por defecto, más información sobre la mayoría de ellas se encuentra en la documentación oficial (http://johnmacfarlane.net/pandoc/README.html/#templates).

Para saber en concreto qué variables hay en la plantilla por defecto podemos también utilizar un filtro Unix como el siguiente:

```
grep -o '\$.*\$' /usr/share/pandoc/data/templates/default.latex \
| grep -v '\$endif\$\|\$else\$\|\$endfor\$'
```

Es especialmente importante destacar que hay una variable crítica pre-definida, la variable `$body$`, que toda plantilla debería incluir, puesto que el contenido propiamente tal de nuestro documento de entrada será introducido en su lugar.

Comprobemos esto último. Creemos una platilla LaTeX **`simple.latex`** con este contenido:

```
\documentclass{minimal}
\begin{document}
$body$
\end{document}
```

y ejecutemos `pandoc` para que reciba el input de la entrada estándar desde un terminal. El resultado de nuestra sesión de prueba es el siguiente:

```
$ pandoc -s --template="simple.latex" --to latex
Hola
Ctrl+D
\documentclass{minimal}
\begin{document}
Hola
\end{document}
```

La primera línea es el comando ejecutado. Estoy pidiendo a `pandoc` que aplique nuestra plantilla, **`simple.latex`** a la entrada que le vamos a pasar y que la convierta a formato LaTeX. Las dos siguientes líneas reproducen lo que he tecleado para que `pandoc` lo consuma. `Ctrl+D` señala `EOF` (fin de fichero) y cierra la entrada estándar. El resto es la salida que `pandoc` produce. Nótese que lo que he tecleado, "Hola", aparece tras el procesamiento, como esperábamos, en el lugar en que estaba `$body$` en la plantilla.

Intentemos algo un poco más complicado. Añadamos una variable de nuestra cosecha, que llamaremos `$saludo$`, a la plantilla:

```
\documentclass{minimal}
\begin{document}
$saludo$
$body$
\end{document}
```

y comprobemos qué pasa:

```
$ pandoc -s -M saludo="Hola gente" --template="simple.latex" --to latex
Esto es Pandoc
Ctrl+D
\documentclass{minimal}
\begin{document}
Hola gente
Esto es Pandoc
\end{document}
```

La orden es casi idéntica a la de antes. El añadido clave es la opción `-M` comentada previamente. A diferencia de la variable predefinida `$body$`, tenemos ahora que pasar los valores de nuestras propias variables a `pandoc` a través de la opción `-M`. Como vemos, en la salida, la variable en la plantilla es sustituida por el valor que hemos dado.

10.2. Condicionales

Los condicionales tienen esta sintaxis:

```
$if(variable)$
X
```

```
$else$
Y
$endif$
```

donde la cláusula `$else$` es opcional.

Supongamos que nos interesa poder elegir, cuando sea necesario, entre diferentes clases de documentos para el mismo documento de entrada. En concreto, supongamos que queremos crear un documento `minimal` por defecto, a no ser que pidamos expresamente que el documento sea de otra determinada clase. Podemos conseguirlo a través de este condicional:

```
$if(mi-clase-doc)$
$mi-clase-doc$
$else$
minimal
$endif$
```

Hay que tener cuidado con la sintaxis. Cada cláusula (`if()`, `else`, `endif`) va rodeada por el signo `$`. Las variables que serán remplazadas por sus valores también van entre `$`. Los valores literales, así como las referencias a la variable en la cláusula `if` van sin ese signo.

Naturalmente, nuestro condicional debe colocarse en el lugar adecuado en la plantilla, a saber, la instrucción `\documentclass`:

```
\documentclass{$if(mi-clase-doc)$
               $mi-clase-doc$
               $else$
               minimal
               $endif$}
```

Escribir lo anterior en una sola línea quizá sea menos legible, pero también más característico del estilo LaTeX. Pongámosla, pues, así:

```
\documentclass{$if(mi-clase-doc)$$mi-clase-doc$$else$minimal$endif$}
\begin{document}
$saludo$
$body$
\end{document}
```

Toca poner a prueba la plantilla:

```
$ pandoc -s -M saludo="Hola gente" --template="simple.latex" --to latex
Esto debería ser minimal
Ctrl+D
\documentclass{minimal}
\begin{document}
Hola gente
Esto debería ser minimal
\end{document}
```

¡Funciona!

Probemos ahora la otra posibilidad:

```
$ pandoc -s -M saludo="Hola gente" -M mi-clase-doc="book" \
--template="simple.latex" --to latex
Y esto, book
Ctrl+D
\documentclass{book}
\begin{document}
Hola gente
Y esto, book
\end{document}
```

¡También funciona! Nótese que en esta ocasión hemos establecido el valor de la variable `mi-clase-doc` a `book` a través de la opción `-M`, tal como hicimos anteriormente con `$saludo$`.

10.3. Bucles

Los bucles funcionan de una manera semejante. La sintaxis básica de un bucle es la siguiente:

```
$for(variable)$
X
$sep$separador
$endfor$
```

La línea `$sep$separador` es opcional. Sirve para definir un separador entre elementos consecutivos.

Digamos que queremos anotar los participantes a una reunión en la primera línea de nuestro documento. Podemos definir una variable `$partipante$` en nuestra plantilla y dejar que Pandoc rellene su contenido. Queremos además que los nombres de los participantes aparezcan separados por una coma. Podríamos hacer todo esto añadiendo lo siguiente a nuestra plantilla:

```
$for(participante)$
$participante$
$sep$,
$endfor$
```

O en una sola línea y en el lugar de la plantilla que corresponde:

```
\documentclass{$if(mi-clase-doc$$mi-clase-doc$$else$minimal$endif$}

\begin{document}
Participantes: $for(participante)$$participante$$sep$, $endfor$
```

```
$saludo$
$body$
\end{document}
```

Hagamos de nuevo una prueba. Ahora añadiremos a la orden `pandoc` tantos `-M participante=nombre-del-participante` como participantes queremos incluir.

```
$ pandoc -s -M saludo="Hola gente" \
-M participante="W. Shakespeare" -M participante="Edgar A. Poe" \
--template="simple.latex" --to latex
Menudo plantel
Ctrl+D
\documentclass{minimal}
\begin{document}
Participantes: W. Shakespeare, Edgar A. Poe

Hola gente
Menudo plantel
\end{document}
```

¡Estupendo! Todo funciona perfecto.

10.4. Bloques de meta-datos

Es, como poco, engorroso tener que pasar todas estas cosas a la línea de órdenes. No es obligatorio, por supuesto. Pandoc proporciona para esta tarea los así llamados *bloques de meta-datos*. Un bloque de meta-datos para nuestro experimento anterior tendría este aspecto:

```
---
mi-clase-doc: minimal
saludo: Hola gente
participante:
- William Shakespeare
- Edgar A. Poe
---
```

Este bloque no es más que un fragmento de texto que sigue las especificación YAML (http://yaml.org/spec). Aparte de esto, los bloques YAML que se incluyen en un documento para que `pandoc` lo procese deben empezar con una línea de tres guiones y terminar con una línea de tres puntos o tres guiones como se ve en el ejemplo.

Un bloque de meta-datos consta de campos. Cada campo tiene un nombre y un valor asociado a ese nombre, separado del nombre por dos puntos. Algunos

campos pueden contener varios valores, los cuales van precedidos por un guion, tal y como aparecen para el campo `participante` en el ejemplo.

Estos bloques nos permiten pasar a pandoc toda la información requerida sin tener que complicarnos la vida con las opciones en la línea de órdenes. La forma habitual de usar estos bloques es añadirlos al principio de nuestro documento de entrada. Otra forma, en mi opinión mejor, es crear un fichero **`yaml`** que pasamos a la vez que el fichero de entrada.

Por ejemplo, si guardamos la entrada de nuestro último experimento (la cadena "Menudo plantel") en un fichero con nombre **`mi_documento.md`** y el bloque de meta-datos en un fichero con nombre **`variables.yaml`**, podemos llamar a `pandoc` como sigue para conseguir exactamente la misma salida que obtuvimos antes:

```
pandoc -s --template="simple.latex" --to latex mi_documento.md variables.yaml
```