Guía *casi* completa de BibTeX*

Joaquín Ataz López

Versión 1, 26 de agosto de 2006

Para sugerencias y correcciones: jal@um.es

Resumen

BibTeX es un programa auxiliar de LaTeX, diseñado para facilitar el manejo de la bibliografía. Pero es más que eso: es también una herramienta que permite a LaTeX extraer datos de una base de datos e insertarlos adecuadamente formateados en un documento.

Este texto pretende abordar todos los aspectos de BibTeX, desde los más básicos a los más complejos, haciendo además hincapié en su *españolización*, pues BibTeX, como tantas otras herramientas, presupone que será usada para trabajar con documentos en inglés.

Índice

Preliminar

BibTeX es un programa que amplia notablemente las capacidades de LaTeX con el manejo de la bibliografía. Las grandes ventajas que ofrece su uso[1] se incrementan cuando se maneja una lista bibliográfica amplia y cuando con la misma bibliografía hay que escribir varios documentos.

No hay mucha documentación sobre BibTeX en español. Publicadas en papel sólo conozco dos obras, ambas de los mismos autores. La primera, [3], sin duda constituye la más completa información sobre BibTeX en nuestro idioma: mas de

[1] BibTeX nos ayuda a gestionar nuestra bibliografía de manera independiente a los documentos en los que debe aparecer, al tiempo que nos permite generar listas de referencias bibliográficas formateadas de manera consistente y cambiar fácilmente determinadas características de las mismas.

40 páginas en las que se incluyen incluso los aspectos más avanzados. La segunda, [4], no es tan extensa, pero constituye una buena introducción en la que se cubren todos los aspectos básicos y algunos aspectos avanzados.

En formato electrónico y en nuestro idioma conozco otros dos documentos. Uno de Luis Seidel, [12] que es una buena introducción, aunque no resulta muy extenso. Otro de José Manuel Mira, [9], que no está pensado para quienes se aproximen por primera vez a esta herramienta, pues en él se explica un *paquete* («`flexbib`») cuyo uso presupone que ya conocemos lo básico del trabajo con BIBTEX.

Como es lógico, en inglés hay mucha más documentación. Los textos principales son el apéndice B del Libro sobre LATEX de Lamport, [7], y los dos documentos que sobre BIBTEX escribió su autor, Oren Patashnik: [10] y [11]. El primero se ocupa de los aspectos más básicos, y está concebido como una *corrección* del mencionado apéndice B, y el segundo se ocupa de los aspectos más avanzados. El problema que tienen estos documentos es que el trabajo de Lamport es difícil de localizar para el público español.

Además de los documentos anteriores, en Internet pueden localizarse varios textos, la mayoría en inglés, algunos introductorios y otros no tanto. De entre ellos me parecen destacables [13] (un documento bastante equilibrado) y [8], dirigido a un público más avanzado que los otros textos mencionados.

***

El objetivo fundamental de este documento es el de, partiendo de la documentación que se acaba de citar, salvo [7] al que no he tenido acceso, y alguna otra que no se ha mencionado todavía, como [6], poner al alcance del público hispanoparlante un documento en el que se aborden todos los aspectos de BIBTEX, desde los más básicos a los más avanzados. De ahí el título que he elegido, que tal vez a alguien le suene a pretencioso. Desde luego esta guía no es completa, porque hay aspectos sobre los que se pasa casi de puntillas[2]. Tal vez ni siquiera sea *casi* completa. Pero sí es la más completa que conozco[3].

La materia a tratar ha sido distribuida en tres partes, atendiendo a su nivel de complejidad y detalle:

1. **BIBTEX básico:** Cubre los aspectos básicos del funcionamiento de BIBTEX que afectan a las instrucciones que hay que incluir en el documento principal de LATEX, al diseño y mantenimiento de la base de datos, y a los distintos

[2]Como por ejemplo las herramientas para gestionar ficheros «`.bib`», numerosos estilos bastante extendidos, o los paquetes de ampliación para trabajar con BIBTEX.

[3]Quizás con la salvedad de [3]. Aunque el problema de ese libro es que hoy no es ya fácil de localizar.

estilos estándar existentes, así como a los procedimientos para *españolizarlos*. Es la parte más extensa del documento. Se supone que sólo con leer esta parte estaremos totalmente preparados para trabajar con BIBTEX.

2. **BIBTEX intermedio:** En esta parte se describen aspectos que no son imprescindibles para trabajar con BIBTEX, pero cuyo conocimiento nos proporcionará un mayor control. En particular se aborda el uso de makebst para generar nuestro propio fichero de estilo.

3. **BIBTEX avanzado:** Esta última parte entra a fondo en los ficheros de estilo de BIBTEX. En ella se describe el lenguaje que estos ficheros usan internamente.

Para entender este documento no hay que ser un LaTEXperto, pero sí hay que saber lo suficiente de LATEX, y del procedimiento estándar que en él se implementa para trabajar con referencias bibliográficas.

En general el contenido de este documento valdrá para cualquier distribución de LATEX que incluya BIBTEX (o sea, todas las distribuciones). Pero hay algunos aspectos que dependen de la concreta instalación que se haya hecho del sistema como, por ejemplo, la localización de los directorios en los que BIBTEX lee las bases de datos o los ficheros de estilo. En tales casos la explicación que doy no vale para cualquier sistema, sino exclusivamente para el mío, que es una instalación estándar de LATEX hecha en un sistema donde funciona Ubuntu versión 6.06. Aunque, en general, estos aspectos dependientes del concreto sistema utilizado son tratados en nota a pie de página.

Asimismo, a lo largo del presente documento se citan en ocasiones determinados paquetes para LATEX o ficheros con estilos adicionales para BIBTEX. Salvo que se diga lo contrario, todos los paquetes y estilos citados están disponibles en la CTAN, («*Comprehensive TeX Archives Network*») y deben buscarse en:

- «http://www.ctan.org/tex-archive/macros/latex/contrib/»
- «http://www.ctan.org/tex-archive/biblio/bibtex/contrib/»

Parte I

BIBTEX básico

1. Dinámica general del sistema

1.1. Preparar nuestro documento principal para usar BIBTEX

Las órdenes bibliography y bibliographystyle:

Para usar BIBTEX necesitamos dos cosas:

1. Tener almacenadas en un fichero aparte, las referencias bibliográficas que pensemos usar.

2. En el lugar del documento principal en el que queramos que aparezca la lista con las referencias bibliográficas, debemos insertar las siguientes dos órdenes de LATEX:

```
\bibliography{MiBiblio}
\bibliographystyle{MiEstilo}
```

   Estas órdenes provocan que tras las oportunas compilaciones (véase la sección 1.2, página 8), se genere una lista con las referencias bibliográficas usadas en el documento en la que, además, a cada una de ellas se le asignará una *etiqueta* identificativa[4]. Los datos correspondientes a tales referencias serán buscados en el fichero «`MiBiblio.bib`», y la lista se formateará de acuerdo con las indicaciones de estilo que se contengan en el fichero de estilo «`MiEstilo.bst`» y se insertará en el lugar del documento en el que se encuentre "`\bibliography`". Las instrucciones de estilo controlarán asimismo el tipo de etiqueta identificativa que se asignará a las obras incluidas en la lista de referencias. Si queremos usar más de una base de datos podemos indicarlas, separadas por comas, como argumento de "`\bibliography`". El

[4] **NOTA TERMINOLÓGICA:** En la documentación en castellano de LATEX el término etiqueta se suele usar para hacer referencia a la cadena de texto que usan en las órdenes que sirven para tratar con referencias cruzadas tales como "`\cite`", "`\label`", "`\index`" o "`\glosary`". Este tipo de *etiquetas* son de uso puramente interno, es decir: LATEX las usa como marcadores para apuntar a distintas partes del documento, pero no se imprimen en él. Junto a estas *etiquetas* también se usa a veces el mismo término para hacer referencia al rótulo identificador de una entrada concreta de la lista de referencias bibliográficas, que se imprime en el documento. Para evitar confusiones, en este documento he reservado el término *etiqueta* para las que llegan a imprimirse en el documento final, mientras que para referirme a las usadas internamente como marcadores, he reservado el término *clave*.

efecto de esto último será que las distintas bases de datos indicadas se concatenerán en el orden en el que se indicaron[5].

Nombres de ficheros y directorios:

El argumento de "`\bibliography`" y el de "`\bibliographystyle`" es el nombre de un fichero. Ambos nombres deben indicarse sin especificar la extensión, pues se asume que esta será la asignada al tipo de fichero de que se trate. De hecho BIBTEX no funciona correctamente cuando el nombre de la base de datos no tiene la extensión «`.bib`», ni tampoco cuando en dicho nombre existe un espacio en blanco o algún carácter no anglosajón como eñes o vocales acentuadas[6].

Los ficheros, el de bibliografía y el de estilo, deben encontrarse en el directorio de trabajo (el mismo en el que se encuentre el documento principal), o en alguno de los directorios en los que LATEX busca por defecto, los cuales varían dependiendo de la distribución de LATEX concreta de que se disponga[7]. Si los ficheros que queremos usar no están en alguno de estos directorios deberemos incluir, junto con el nombre del fichero, la ruta de acceso al mismo, en cuyo caso para dicha ruta se aplican las mismas reglas que para el nombre de los ficheros: no podrá contener espacios en blanco ni caracteres no anglosajones.

Contenido de la lista de referencias a generar:

Las características *formales* y de ordenación de la lista bibliográfica que se generará, dependen del estilo concreto usado, y se explican en la sección 3. Aquí me concentraré exclusivamente en los siguientes dos aspectos de la misma:

---

[5] Lo que tiene importancia, porque en los ficheros «`.bib`» hay utilidades que sólo funcionan si el orden en el que ciertos elementos se encuentran dentro del fichero es el correcto. Por ejemplo, las abreviaturas deben estar definidas antes de ser usadas, de modo que si tenemos un fichero que recoja una serie de abreviaturas para BIBTEX y una base de datos que las use, siempre habrá que indicar el fichero de abreviaturas antes que la base de datos.

[6] Así ocurre en mi sistema. Es posible que en otros sistemas operativos no exista este problema, aunque creo que ocurrirá lo mismo, al menos en lo relativo a la extensión de los ficheros; porque "`\thebibliography`" se limita a copiar en el fichero «`.aux`» el texto que recibe como argumento, y BIBTEX le añade a dicho texto la extensión «`.bib`», salvo en el caso de que ya termine así.

[7] En «`web2c`», que es la distribución de LATEX estándar para sistemas Unix/Linux, el árbol de ficheros de TEX se encuentra en «`/usr/share/texmf`» Allí hay un fichero denominado «`texmf.cnf`» en el que se almacenan las variables que controlan las rutas de búsqueda por defecto de LATEX y sus programas auxiliares. Aunque a veces dicho fichero se encuentra en el directorio «`web2c`», o en «`/etc/texmf/`». En dicho fichero se encuentran las dos variables que afectan a los directorios relativos a BIBTEX: BIBINPUTS (directorios en los que se buscarán los ficheros «`.bib`») y BSTINPUTS (directorios en los que se buscarán los ficheros de estilo).

Qué referencias se incluyen en la lista: En ella se contendrán todas las referencias bibliográficas que, a lo largo de nuestro documento, se hayan referenciado mediante los comandos "`\cite`" y "`\nocite`":

- El comando "`\cite[DatosAdicionales]{clave}`" produce un doble efecto. En primer lugar, la referencia bibliográfica identificada en la base de datos mediante la clave recibida como parámetro, se incluirá en la lista bibliográfica. En segundo lugar, en el punto del documento donde se encontrara el comando, se imprimirá la etiqueta asignada a tal referencia en la lista de referencias junto con los datos adicionales que eventualmente hayamos incluido en el argumento opcional del comando.
- El comando "`\nocite{clave}`" produce el primero de los efectos indicados, pero no el segundo, es decir: la obra identificada por la clave será incluida en la lista bibliográfica final, y en ella se le asignará asimismo una etiqueta (como a todas las obras de la lista), pero en el lugar del documento en el que se encuentra el comando "`\nocite`" no se imprimirá nada.

Ambos comandos pueden recibir, como parámetro, las claves de varias referencias distintas, separadas mediante comas. Y en el caso concreto de "`\nocite`", si se escribe "`\nocite{*}`" el efecto será incluir en la lista bibliográfica final todas las referencias bibliográficas incluidas en la base de datos.

Qué título tendrá la lista de referencias: Dependiendo del tipo de documento de que se trate, el título de la lista de referencias está controlado por dos variables de nombre distinto[8]. En documentos tipo «`book`» para cambiar el título de la lista debe modificarse el comando "`\bibname`", mientras que en documentos tipo «`article`» el comando LaTeX que hay que modificar es "`\refname`". El paquete «`babel`», por otra parte asigna a cada uno de estos comandos un valor dependiente del idioma. En español los nombres son "Bibliografía" para los libros y "Referencias" para los artículos.

Si, por ejemplo, deseamos que la lista de referencias se titule "Bibliografía citada" y nuestro documento es de tipo «`book`» tendremos incluir en el cuerpo de nuestro documento[9] la siguiente orden:

```
\renewcommand{\bibname}{Bibliograf\'ia citada}
```

[8] Lo que, en mi opinión, es una inconsistencia del sistema. El paquete «`chbibref`» incluye un comando que permite cambiar el título de la lista de referencias con independencia de que se trate de un documento tipo «`book`» o de un documento tipo «`article`».

[9] Y es que, según mi experiencia, esta orden no produce efectos si se incluye en el preámbulo. Posiblemente por la interacción que la carga del paquete «`babel`» produce.

Asimismo, si se desea que en el índice de contenido de nuestro documento aparezca una entrada para la bibliografía, hay que incluirla a mano mediante la orden "`\addscontentline`" de LaTeX[10], la cual debe insertarse *antes* de "`\bibliography`", ya que si se inserta detrás, en el índice del documento aparecerá como página de la bibliografía la página en la que esta termina, y no la página en la que empieza[11].

1.2. Cómo generar la lista de referencias bibliográficas

Una vez que nuestro documento (llamémosle «`MiDoc.tex`») está preparado, en los términos que se acaban de exponer, debemos seguir los siguientes pasos:

1º Compilar con LaTeX nuestro documento «`.tex`». Ello hará que se genere un fichero de extensión «`.aux`», en el que se incluirá información sobre la base de datos a usar, el fichero de estilo a usar, y las referencias bibliográficas que hay que incluir en la lista de referencias.

2º Ejecutar, desde la línea de comandos[12], la orden «`bibtex MiDoc`». Ello hará que BibTeX lea el fichero «`.aux`»[13], generado por la anterior compilación del documento, extrayendo de él la información que necesita para trabajar: qué base de datos debe usar, qué estilo, y qué referencias hay que buscar en la base. Y así, tras extraer de la base de datos los registros precisos, y formatearlos de acuerdo con el estilo indicado, BibTeX genera un fichero de extensión «`.bbl`» en el que se contienen los comandos de LaTeX necesarios para escribir la lista de referencias bibliográficas que hay que insertar en el documento principal. BibTeX genera también un fichero adicional, de extensión «`.blg`» que es un fichero «`.log`»[14].

3º Compilar de nuevo el documento principal con LaTeX. En esta segunda compilación, al leer la orden "`\bibliography`", se insertará en su lugar el contenido

---

[10] El paquete «`tocbibind`» hace que los índices y la lista de referencias se incluyan automáticamente en la tabla de contenido.

[11] Véase más adelante en la sección 6.2, a propósito de los elementos «`@preamble`» de los ficheros «`.bib`», cómo podemos incluir en un fichero «`.bib`» el código necesario para cambiar automáticamente el título a la lista de referencias y asegurarnos de que se incluye en la tabla de contenido.

[12] Numerosas herramientas para LaTeX incorporan la posibilidad de ejecutar desde el mismo editor tanto LaTeX como cualquiera de sus programas auxiliares, en cuyo caso no sería preciso ejecutar BibTeX *desde la línea de comando*.

[13] De hecho BibTeX se ejecuta sobre el fichero «`.aux`», y no sobre el fichero «`.tex`». Este detalle puede escapársenos porque BibTeX no exige que se escriba la extensión. Pero si en el comando incluimos la extensión comprobaremos que «`bibtex MiDoc.tex`» generaría un error, mientras que «`bibtex MiDoc.aux`» funcionaría correctamente.

[14] Es decir: en él se almacenan todas las salidas generadas por BibTeX, incluyendo los mensajes de advertencia o error. En caso de que algo no haya funcionado como esperábamos es imprescindible la consulta de este fichero para ver qué es lo que ha podido fallar.

del fichero «`.bbl`» generado en el paso anterior. Asimismo la nueva compilación reescribe el fichero «`.aux`», añadiendo a la información que ya existía en él la generada ahora, que es más completa pues incluye los datos de la lista bibliográfica final que se acaba de insertar en el documento.

4º Esta última información es usada en una nueva compilación con LaTeX para escribir correctamente los rótulos que hay que colocar en lugar de los comandos "`\cite`". Y eventualmente puede ser necesaria una nueva compilación: cuando alguno de los campos de la base de datos contenga algún comando de LaTeX que implique el uso de referencias cruzadas. En particular, el comando "`\cite`".

¿Parece complicado? No lo es. Lo que ocurre es que lo he explicado incluyendo detalles sobre cómo interactúan entre sí los distintos ficheros generados durante las compilaciones de LaTeX. Simplificando y sin entrar en tales detalles, lo que hay que hacer es, tras haber indicado en el documento principal el nombre de la base de datos y del estilo, realizar una primera compilación con LaTeX, ejecutar BibTeX y luego compilar de nuevo con LaTeX dos veces (o, en ciertos casos, tres). Esta doble (o triple) compilación final no es ninguna especialidad de BibTeX, sino que es requerida por LaTeX siempre que trabaja con referencias cruzadas. Asimismo habrá que repetir el proceso cada vez que incorporemos a nuestro documento principal un comando que suponga una modificación de lista bibliográfica final, bien por cambiarle el estilo, bien por incluir o eliminar referencias bibliográficas.

2. Las bases de datos bibliográficas

2.1. Reglas generales sobre la escritura y codificación de los ficheros «`.bib`»

Los ficheros «`.bib`» usados por BibTeX son ficheros de texto sometidos a las siguientes reglas generales:

1. El tratamiento de los espacios en blanco, tabuladores y saltos de línea es similar al que estos caracteres reciben en LaTeX[15]. Son simples delimitadores de palabras y la regla es que da igual cuántos delimitadores haya entre dos palabras[16].

[15] Hay, no obstante, una diferencia, y es que en LaTeX una línea en blanco es relevante pues sirve para indicar un cambio de párrafo. En los ficheros «`.bib`», una línea en blanco es tan irrelevante como un salto de línea simple.

[16] Podríamos, en teoría, escribir todo el fichero «`.bib`» como una sola e inmensa línea en la que las palabras estuvieran separadas entre sí por un solo espacio en blanco. Pero tal fichero sería, para los seres humanos, más difícil de leer que otro en el que hayamos usado los tabuladores, espacios y saltos de línea para mejorar el aspecto visual (llamado *legibilidad*) del fichero.

2. BIBTEX no distingue entre mayúsculas y minúsculas[17].

Uso de caracteres españoles en los ficheros «`.bib`»

La versión original de BIBTEX no era capaz de trabajar con caracteres no anglosajones (como eñes o vocales acentuadas). En la actualidad existe una versión de BIBTEX que sí puede hacerlo. No obstante se suele recomendar que para representar esos caracteres que requieren 8 bits se sigan las reglas generales de LATEX y así, por ejemplo, en lugar de escribir «`Pág`» escribamos "`P\'ag`"[18]. Y aunque es cierto que escribir de ese modo nuestro idioma es incómodo, hay herramientas que lo facilitan[19].

En todo caso, si decidimos introducir caracteres no anglosajones directamente en el fichero «`.bib`», debemos tener en cuenta que como el contenido de este fichero puede eventualmente incorporarse a un documento «`.tex`», si en ambos el tipo de codificación no coincidiera se generaría un error. Por ello, para usar estos caracteres deben cumplirse los siguientes requisitos:

a) Debemos guardar el fichero «`.bib`» con la misma codificación con la que vayamos a guardar el fichero «`.tex`» en el que se hará uso de la base de datos.

b) En el fichero «`.tex`» debe usarse la orden "`\usepackage[cod]{inputenc}`" donde "cod" significa una cadena representativa de la codificación usada en el fichero «`.bib`» (y en el «`.tex`»): «`utf8`», «`latin1`», etc.

c) Debemos asegurarnos de que estamos usando la versión de BIBTEX de 8 bits.

2.2. Registros y campos bibliográficos

Un fichero «`.bib`» almacena una base de datos bibliográfica. En toda base de datos hay dos nociones fundamentales:

---

[17] La no distinción entre mayúsculas y minúsculas se aplica a los elementos de BIBTEX (nombres de registros o campos, abreviaturas, etc), pero no al contenido de los campos de los registros.

[18] El problema de representar caracteres no anglosajones directamente en un fichero de texto, es que estos caracteres no están incluidos en la tabla ASCII que todos los ordenadores reconocen, sino en otras tablas o *codificaciones*. Y, desgraciadamente para los hablantes de idiomas distintos al inglés, existen *muchas* codificaciones distintas para representar los mismos caracteres: "utf8", "latin1", "cp850", etc.; y para trabajar con un documento que contenga este tipo de caracteres es preciso saber exactamente qué codificación se usó para almacenarlo pero, desgraciadamente, los ficheros de texto no guardan información sobre su propia codificación. Además, en el caso concreto de los ficheros «`.bib`», el uso de una codificación u otra puede afectar a la ordenación alfabética de los registros almacenados en la base de datos.

[19] Como, por ejemplo «`pybliographer`», una utilidad de gestión de ficheros «`.bib`» para sistemas Unix/Linux, en la que podemos teclear normalmente cualquier carácter, pero antes de guardar el fichero, los caracteres no anglosajones son convertidos a la secuencia LATEX necesaria para representarlos. De esta herramienta se vuelve a hablar en la sección 2.4.1.

Campo: Un dato aislado y básico. Por ejemplo, el año de edición de un libro, o el nombre del autor de un artículo de revista, o la página del libro de las actas de un congreso en donde empieza una determinada ponencia...

Registro: Un conjunto de campos que permiten describir de forma completa un ejemplar del tipo de realidades a que se refiere la base de datos. En nuestro caso, tratándose de una base de datos bibliográfica, cada registro describe una referencia bibliográfica concreta (y completa).

Es decir: todos los campos que contienen datos de una misma referencia bibliográfica, constituirán un registro. Y la base de datos, en sí misma considerada, no será sino un conjunto de registros, cada uno de los cuales consta de varios campos.

2.2.1. Formato general de los registros y los campos

En los ficheros «`.bib`» se usa el carácter "@", seguido de una palabra representativa del tipo de registro de que se trate, para indicar que empieza un registro. Asimismo se usan llaves para delimitar el contenido del registro. En consecuencia el formato de un registro es el siguiente:

```
@TipoRegistro{clave,
   Campo1,
   Campo2,
   ...
   CampoN,
}
```

Donde *TipoRegistro* es el nombre que identifica a un concreto tipo de registro, y puede escribirse indistintamente en mayúsculas o minúsculas. En el contenido del registro, hay básicamente dos elementos:

La clave de identificación del registro es siempre su primer elemento. Se usa para distinguir a un registro concreto del resto de los registros de la base de datos, y por ello en una misma base de datos no debe haber dos registros que tengan claves iguales, y tampoco deben unirse o manejarse en un mismo documento dos o más bases de datos si ello implica que alguna clave vaya a repetirse[20]. La clave puede consistir en cualquier combinación de letras, números y ciertos signos de puntuación. No se admiten ni los caracteres no

[20] Digo que no *deben* usarse claves iguales; no que no pueda hacerse. Ello es porque BIBTEX no genera ningún error cuando en un fichero dos o más registros tienen la misma clave. Simplemente, en tal caso, toma en consideración exclusivamente la primera aparición de la clave, ignorando las restantes. Y, a tales efectos, téngase en cuenta que para el control de las claves BIBTEX no distingue entre mayúsculas y minúsculas.

anglosajones (vocales acentuadas y los caracteres “¡¿ñÑçÇ”) ni ciertos signos de puntuación como la coma (que se usa para indicar que la clave ha terminado).

En las citas hechas en el documento «`.tex`» habrá que usar esta clave para identificar al registro al que se quiere hacer referencia.

Los campos: A BIBTEX le es indiferente en qué orden se escriban los campos de un determinado registro, así como si se usan cero, uno o más saltos de línea entre dos campos.

Para escribir un campo dentro de un registro podemos usar indistintamente cualquiera de los dos formatos siguientes:

```
NombreCampo = { Contenido },
NombreCampo = " Contenido ",
```

En ambos casos intervienen los siguientes elementos:

Nombre del campo: Normalmente será uno de los campos, obligatorios u opcionales, previstos para el tipo de registro de que se trate. Si el nombre no coincide con ninguno de los que el estilo ha previsto para el tipo de registro que sea, su contenido será ignorado. Si el mismo campo se introduce más de una vez en el mismo registro, sólo será tomada en consideración su primera aparición.

Delimitadores de contenido: Se exige el uso de delimitadores para el contenido del campo siempre que este no tenga un valor puramente numérico o consista en una abreviatura previamente definida como tal mediante la función «`@String`» (véase la sección 2.3.1). Como delimitadores se pueden usar las llaves o las comillas dobles. La diferencia entre usar unas u otras es sólo una: cuando usamos comillas dobles como delimitadores, dentro de la cadena delimitada no podemos usar comillas dobles, salvo que las encerremos entre llaves. Por ejemplo:

```
title = {Comentarios a "El Buscón" de Quevedo},
title = "Comentarios a {"}El Buscón{"} de Quevedo",
```

servirían para incluir un título que a su vez incluya dobles comillas.

El contenido de los campos: cualquier cosa que se incluya en un campo puede, eventualmente, incorporarse a un documento LATEX, y por lo tanto:

1. Es posible incluir, como contenido de un campo, comandos de LATEX, aunque ello sólo está recomendado en casos muy especiales, y nunca es una buena idea que esos comandos sean de *formato* de texto, pues la idea que está detrás de BIBTEX es que el formateo de las listas de referencias se haga depender del estilo usado.

2. Dentro del contenido del campo hay que respetar las reglas generales de LaTeX, no usar sus caracteres reservados (salvo cuando se trate de un uso legal) y recordar que LaTeX, a diferencia de BibTeX, sí distingue entre mayúsculas y minúsculas.
3. Las llaves que pueda haber en un campo deben estar equilibradas. Es decir: toda llave abierta debe ser cerrada. Y ello se aplica incluso para llaves que se deben imprimir[21].

Existen reglas especiales para el contenido de algunos campos concretos, pero estas se exponen más adelante, cuando se explican los campos reconocidos por los estilos estándar de BibTeX.

La coma final: Indica que el contenido del campo ha terminado y *puede* empezar otro campo distinto. No es obligatoria en el último campo de un registro; pero no se produce ningún error por el hecho de introducirla también en él.

Desde el punto de vista de un concreto estilo bibliográfico, los campos presentes en un registro se incluirán en uno de los siguientes grupos:

Campos obligatorios: Son aquellos que el estilo en cuestión espera encontrar en un determinado tipo de registro, de manera que si en un registro concreto alguno de ellos no está presente, se generará una advertencia en la salida estándar y en el fichero «`.blg`» generado por BibTeX. En algunas ocasiones la falta de uno de estos campos puede provocar un error, aunque eso no debería ocurrir en un estilo bien diseñado.

Campos opcionales: Son aquellos para los que el estilo en cuestión ha previsto algún tipo de acción, pero se ha previsto también su posible ausencia, y esta no tiene especial trascendencia.

Campos ignorados: Cualquier otro campo presente para el que el estilo en cuestión no ha previsto ningún tipo de acción.

2.2.2. Tipos de registros bibliográficos

Las referencias bibliográficas pueden ser de distinta naturaleza, y en consecuencia se distinguen diferentes tipos de registros. A continuación expondré los nombres de los distintos tipos de registro previstos en BibTeX, indicando para cada uno de ellos sus campos obligatorios y opcionales específicos[22]. Para que

[21] Lo que se debe a que BibTeX no interpreta los comandos de LaTeX, por lo tanto si en un campo se lee el texto «`\{`», BibTeX no ve un caracter imprimible, sino una llave abierta que debe ser cerrada dentro del campo. Por ello en el raro caso de que debamos incluir como contenido de un campo el carácter "{" (o "}") habría que usar el comando LaTeX "`\leftbrace`" (o "`\rightbrace`").

[22] Con *específicos* quiero decir que, para evitar repeticiones, en la relación que sigue no se ha incluido en cada uno de los registros los nombres de dos campos opcionales que están presentes en todos los registros, sean del tipo que sean: «`key`» y «`note`».

BIBTEX pueda reconocer adecuadamente los registros y campos, los nombres de estos deben ser usados en inglés, y por ello los mencionaré en ese idioma. Entre paréntesis añadiré, para los tipos de registro, lo que significa su nombre. En la próxima sección se explica para qué sirve cada campo.

Article (artículo): Un artículo publicado en una revista. Campos obligatorios: `author`, `title`, `journal` y `year`. Opcionales: `volume`, `number`, `pages` y `month`.

Book (libro): Un libro *normal*. Campos obligatorios: `author` o `editor`, `title`, `publisher` y `year`. Opcionales: `volume` o `number`, `series`, `address`, `edition` y `month`.

Booklet (folleto): Un trabajo impreso y distribuido pero sin que conste la editorial o institución que lo patrocina. Campo obligatorio: `title`. Campos opcionales: `author`, `howpublished`, `address`, `month` y `year`.

Conference (conferencia): Idéntico a InProceedings. Se incluye exclusivamente para mantener la compatibilidad con el formato *Scribe*.

InBook (dentro de un libro): Una parte de un libro, que puede ser un capítulo (o sección o similar) o un rango de páginas, o ambas cosas. Campos obligatorios: `author` o `editor`, `title`, `chapter` y/o `pages`, `publisher` y `year`. Opcionales: `volume` o `number`, `series`, `type`, `address`, `edition` y `month`. El campo `title`, en estas referencias, se refiere al título del libro, no al título del capítulo o grupo de páginas a que se refiere el registro.

InCollection (en una colección): Una parte de un libro que tiene su propio título. Campos obligatorios: `author`, `title`, `booktitle`, `publisher` y `year`. Opcionales: `crossref`, `editor`, `volume` o `number`, `series`, `type`, `chapter`, `pages`, `address`, `edition` y `month`.

InProceedings (en las actas): Una conferencia, artículo o ponencia en las actas de un congreso o, o, en general, en un libro que agrupe varios trabajos de autores distintos y con títulos independientes. Campos obligatorios: `author`, `title`, `booktitle`, `year`. Campo opcionales: `crossref`, `editor`, `volume` o `number`, `series`, `pages`, `address`, `month`, `organization` y `publisher`.

Manual: Documentación técnica. Campo obligatorio: `title`. Campos opcionales: `author`, `organization`, `address`, `edition`, `month` y `year`.

MasterThesis (Proyecto fin de carrera): Lo que en el mundo académico norteamericano se denomina "*Master Thesis*" o "*Minor Thesis*" y que en España equivale a las llamadas "*Tesinas de licenciatura*" y a los *Proyectos de fin de carrera*; es decir: un trabajo de investigación menor, leído en una institución académica. Campos obligatorios: `author`, `title`, `school` y `year`. Opcionales: `type`, `address` y `month`.

Misc (miscelánea): Este tipo está previsto para ser usado cuando ninguno de los otros tipos encaje bien. Su peculiaridad es que carece de campos obligatorios. Campos opcionales: `author`, `title`, `howpublished`, `month` y `year`.

PhdThesis (tesis doctoral): Lo que en el mundo académico norteamericano se denomina "*PhD Thesis*", "*Major Thesis*" y que equivale a las tesis doctorales españolas, es decir: un trabajo de investigación original y extenso que permite obtener el título de doctor que constituye la más alta graduación académica existente. Campos obligatorios: `author`, `title`, `school` y `year`. Opcionales: `type`, `address` y `month`.

Proceedings (libro de actas): El libro de actas donde se encuentra una conferencia o ponencia en un congreso, aunque yo lo uso con carácter general para todos aquellos libros que agrupan trabajos, cada uno de ellos con distintos autores. Campos obligatorios: `title` y `year`. Opcionales: `booktitle`, `editor`, `volume` o `number`, `series`, `address`, `month`, `organization` y `publisher`.

TechReport (informe técnico): Un informe publicado por un centro académico o institución similar, normalmente numerado dentro de una serie. Campos obligatorios: `author`, `title`, `institution` y `year`. Campos opcionales: `type`, `number`, `address` y `month`.

Unpublished (no publicado): Un documento que tiene un autor y un título pero que formalmente no ha sido publicado. Campos obligatorios: `author`, `title` y `note`. Opcionales: `month` y `year`.

A los campos mencionados hay que añadir, en todos los tipos de registro, un campo `key` y un campo `note`, ambos opcionales, salvo en las referencias tipo «`Unpublished`», donde `note` es obligatorio.

2.2.3. Significado de los distintos campos bibliográficos

Address: Este campo está pensado para almacenar la dirección de la editorial o institución responsable de una publicación. Es siempre opcional y conviene usarlo, sobre todo, cuando las editoriales sean pequeñas o poco conocidas. En la mayor parte de los casos, además, no se incluye aquí la dirección completa, sino exclusivamente la ciudad donde reside la editorial. Hay especialidades científicas donde es costumbre identificar las obras publicadas más bien por la ciudad que por la editorial, en cuyo caso puede ser buena idea indicar la ciudad no en el campo «`address`», sino en «`publisher`».

Author: Como este es el campo más complejo, dividiré su explicación en varios apartados:

La autoría de las referencias: Toda referencia bibliográfica ha de tener una *autoría* y en tal sentido, en todos los tipos de registros hay algún campo dirigido a recogerla. Ese campo es normalmente «`author`», aunque:

a) En «`Proceedings`» la autoría viene recogida por el campo «`editor`», porque este tipo de registro recoge un libro sin autor propiamente dicho, que recopila o agrupa varios trabajos que sí tienen autor.
b) En los tipos «`Book`» e «`InBook`», el campo «`author`» puede ser sustituido por «`editor`». Si se especifican ambos campos simultáneamente, el contenido del campo «`editor`» es ignorado.
c) En las referencias de tipo «`Manual`», a falta de campo «`author`», la autoría se atribuirá al campo «`organization`».

En cualquiera de estos casos, la recomendación es que el nombre del autor se escriba tan completo como se conozca, es decir: que no se usen iniciales para sustituir al nombre propio, ya que ese efecto podemos obtenerlo mediante el estilo bibliográfico. Pero si en la base de datos no consta el nombre completo no hay estilo bibliográfico que sea capaz de adivinarlo.

Varios autores: Si hay más de un autor, para separar unos de otros hay que usar necesariamente la palabra "and" para que BIBTEX sepa donde empieza un autor y donde acaba otro. Si usamos la conjunción española equivalente, "y", BIBTEX no formateará correctamente el contenido de este campo porque no sabrá separar a los distintos autores. El hecho de usar "and" dentro del fichero «`.bib`» no significa que en nuestros documentos se vaya a usar dicha palabra, pues eso depende del fichero de estilo.

Si *dentro* del nombre de uno de los autores aparece la palabra "and", hay que encerrar entre llaves (distintas de las genéricas llaves para delimitar el contenido del campo) el nombre completo. Así, en los siguientes ejemplos:

```
author = "{McArthur And Inc.}",
author = {{McArthur And Inc.}},
author = {Thomas R. Delan And {McArthur And Inc.}},
```

El nombre que contiene, como parte de él, la palabra "And" va siempre entre llaves, con independencia de qué tipo de delimitador se haya escogido para el campo y de si hay o no mas nombres.

Distinción entre nombre y apellidos: El nombre de una persona tiene varias partes de las que las principales (en los nombres españoles) son el nombre propio (también llamado nombre de pila, o nombre a secas) y los apellidos. BIBTEX descompone los nombres para determinar qué parte de

los mismos hay que atribuir al nombre propio, cuál a los apellidos, y cual a las restantes partes que BIBTEX reconoce en un nombre[23]. Para ello se siguen las siguientes reglas:

1ª. Si el texto introducido consta de una sola palabra, se asignará al apellido.

2ª. Si consta de dos o más palabras, se considerará que la última es el apellido y el resto es el nombre de pila. Esta regla se basa en que fuera de la Península Ibérica (y paises de tradición cultural ibérica), es usual tener un solo apellido y varios nombres de pila.

3ª. En el caso de que el apellido conste de varias palabras y alguna de las palabras interiores estuviera escrita con minúsculas, se considerará que tal palabra es una partícula que separa el nombre del apellido por lo que se asignará al apellido todo lo que esté detrás de tal partícula, y al nombre lo que esté antes. Y así, por ejemplo, BIBTEX interpretará correctamente el nombre

   ```
   Miguel de Cervantes Saavedra
   ```

   siempre y cuando escribamos el "de" y lo hagamos en minúsculas. Pero si escribimos:

   ```
   Miguel Cervantes Saavedra
   Miguel De Cervantes Saavedra
   miguel de cervantes saavedra
   ```

   como BIBTEX no podrá identificar la partícula aplicará la regla 2, y considerará que "Saavedra" es el apellido y el resto nombre propio[24].

4ª. Si solo conocemos el apellido, y este consta de varias palabras, para evitar que las primeras se asignen al nombre de pila, hay que encerrar todo el nombre entre llaves (además de las delimitadoras del campo). Por ejemplo:

   ```
   author = "{Ortega y Gasset}",
   ```

5ª. Si en un nombre se usa una coma, BIBTEX asignará al apellido todo lo que esté antes que la coma, y al nombre propio todo lo que esté detrás. Esta regla prevalece sobre todas las anteriores.

---

[23] En realidad BIBTEX descompone los nombres en cuatro partes: Nombre de pila (en inglés, *First name)*, Partícula de separación, Apellidos (en inglés *Last name* o *Surname* y partícula "Jr.", muy habitual en los nombres anglosajones. En las líneas que siguen he omitido lo referente a esta última partícula, inexistente fuera del mundo anglosajón. No obstante si en nuestra base de datos debe introducirse algún nombre propio que incluya dicha partícula, normalmente esta será reconocida por BIBTEX y su presencia no alterará las reglas que a continuación se exponen.

[24] Por ello si queremos que en nuestra lista bibliográfica las partículas se impriman en mayúsculas, pero que BIBTEX las reconozca correctamente, habría que escribir: "`Miguel {\uppercase{d}e} Cervantes Saavedra`".

Para nombres españoles, con dos apellidos, lo más seguro es usar una coma entre el apellido y el nombre para impedir que BIBTEX realice malas interpretaciones. Porque aunque en ocasiones la partícula nos puede ayudar (como en "Miguel de Cervantes Saavedra") la mayoría de los nombres carecen de partícula, y en muchas ocasiones la partícula produciría un error, como en "Félix Lope de Vega y Carpio".

Booktitle: Es el título de un libro, parte del cual está siendo citado. Se usa, por lo tanto en las referencias que sirven para citar una parte de un libro que tenga un título distinto del título del libro, es decir: «`InCollection`» e «`InProceedings`». Este campo también está presente en las referencias de tipo «`Proceedings`», aunque en ellas se usa exclusivamente para las referencias cruzadas (véase la sección 2.3.2). A este campo le son de aplicación las reglas que más adelante se exponen sobre el campo «`title`».

Chapter: Contiene un número de capitulo, sección o unidad estructural de un libro. Se usa en las referencias de tipo InBook e InCollection para identificar una parte de un libro.

Crossref: Contiene una etiqueta en la base de datos para generar referencias cruzadas internas (véase la sección 2.3.2).

Edition: Se trata del número de edición de un libro. Es siempre un campo opcional; y de hecho sólo suele rellenarse cuando no se trata de la primera edición. Los estilos estándar presuponen que se escribirá con una palabra en mayúsculas (y en inglés). Por ejemplo "Second", aunque si se escribe en español ("Segunda"), o con un ordinal expresado en forma numérica ("2ª"), no pasa nada[25]. Lo que sí es importante, para generar listas bibliográficas consistentes, es que siempre lo escribamos del mismo modo.

Editor: Este campo ya ha sido mencionado a propósito del campo "author". El editor de un libro es el que, sin ser su autor propiamente dicho, lo impulsa y hace nacer. Esta figura es importante hasta el punto de que sobre él recae la *autoría* (a efectos de la base de datos) en determinado tipo de obras como puede ser una antología, un estudio crítico de alguna obra antigua, etc. Por ejemplo: una edición crítica de la poesía barroca española, o el conocido "*Las mil mejores poesías de la lengua castellana*". Las reglas relativas a este campo son similares a las del campo "author".

Howpublished: Este campo se usa en las referencias de tipo "Booklet" y "Misc" que recogen libros que no han sido publicados por una editorial o institución "normal". En este campo puede indicarse cómo se llegó a publicar la obra de que se trate. La primera palabra debe ir en mayúsculas.

[25] En los estilos estándar detrás de este campo se añade la palabra "edition" y, si están españolizados, la palabra "edición", por ello no conviene usar directamente un número que no sea un ordinal, ya que entonces la referencia contendría la expresión "2 edición", que no es correcta.

Institution: Este campo existe exclusivamente en los informes técnicos (referencias «`TechReport`») y recoge el nombre de la institución que ha financiado el informe de que se trate.

Journal: Para artículos publicados en revistas este campo recoge el nombre de la revista. En determinados campos del conocimiento existen abreviaturas estándar para las revistas más conocidas, por lo que puede usarse la abreviatura en lugar del nombre completo de la revista. Sobre el uso de abreviaturas en los ficheros «`.bib`» véase la sección 2.3.1.

Key: Está disponible como campo opcional en todos los tipos de referencias. Se usa en relación con la generación de etiquetas, en aquellos estilos en los que estas se generan a partir del nombre del autor, como por ejemplo ocurre en el estilo «`alpha`». En tales casos, el valor del campo «`key`» será usado para la generación de la etiqueta si no consta ningún valor para el campo «`author`», incluso aunque si conste algún valor para alguno de los campos que en ciertos tipos de referencias pueden suplir a dicho campo (véase lo que, al explicar el campo «`author`» se dijo sobre la autoría, en pág. 16).

Month: En artículos de revista se usa para designar el mes en el que salió el número que contiene dicho artículo. En otro tipo de trabajos el mes en el que fueron publicados o terminados (para trabajos no publicados). La documentación oficial de BIBTEX recomienda introducir este campo mediante unas abreviaturas estándar en inglés[26], en cuyo caso usando la abreviatura, no habría que usar delimitadores para el campo (por las razones que se explican cuando se habla de las abreviaturas en la sección 2.3.1).

En teoría aquí ocurre igual que en el campo «`author`», respecto de la partícula "and" usada para separar entre sí a los distintos autores, y es que aunque hayamos introducido el mes en inglés (o mediante una abreviatura del nombre inglés), luego, usando el estilo bibliográfico correcto, en las referencias que se inserten en nuestros documentos aparecerá el nombre en español. Aunque, a diferencia de lo que ocurre con el campo «`author`», si aquí introducimos la información directamente en español, no se produce ningún error.

Note: Cualquier tipo de información adicional que no tenga cabida en el resto de los campos[27]. En los estilos bibliográficos estándar el contenido de este

[26] Se trata de: «`jan`», «`feb`», «`mar`», «`apr`», «`may`», «`jun`», «`jul`», «`aug`», «`sep`», «`oct`», «`nov`» y «`dec`».

[27] Yo lo he usado, por ejemplo, para recoger el nombre de los traductores (en traducciones), o en ediciones que han sido actualizadas por personas distintas del autor original, para recoger el nombre de estos últimos, o en documentación electrónica, para recoger la dirección de Internet donde se encuentra. Aunque para esto último es preferible usar un campo adicional de nombre «`url`», pues existen varios estilos no estándar para BIBTEX que preven la existencia de este campo y son capaces

campo se transcribe al final de la referencia y su primera palabra debería empezar en mayúsculas.

Number: Puede tener distintos significados. En los informes técnicos recoge el número de informe de que se trate. En artículos de revista se usa en revistas cuyos distintos ejemplares se identifiquan mediante un *número*, o mediante un número y un volumen. En libros sólo se usa si el libro forma parte de una serie o colección. En los registros de tipo «`Book`» y «`Proceedings`» (así como en sus derivados: «`InBook`», «`InCollection`» e «`InProceedings`») el contenido de este campo sólo es tomado en consideración si el campo «`volume`» se ha dejado en blanco.

Organization: La organización que financia una conferencia o que publica un manual. En los registros de tipo «`Manual`» este campo designa la *autoría* si «`author`» carece de contenido.

Pages: Se usa en artículos y en partes de libro para designar las páginas concretas en las que un trabajo se encuentra. Los estilos estándar de BIBTEX esperan que se pongan aquí las páginas inicial y final, separadas por uno o dos guiones. De hecho estos estilos convierten el guión sencillo en uno doble, que es el utilizado en LATEX para designar intervalos de páginas. Si se quiere indicar sólo la primera página, se utiliza el carácter "+" que significa "desde esa página en adelante". Como los estilos estándar suponen que se trata de más de una página, en la lista bibliográfica final, a veces al número aquí puesto se le añade la palabra "páginas" (*pages*), o la abreviatura "págs." (pp) en plural.

Publisher: La editorial encargada de la publicación de un libro. Su contenido suele complementarse con el del campo «`address`».

School: Nombre de la Escuela, Facultad o Instituto en donde se ha confeccionado un trabajo académico (tesis doctoral o tesina de licenciatura).

Series: En libros que forman parte de una colección, se usa para recoger el nombre de la misma.

Title: Es un campo presente en todos los tipos de referencias bibliográficas, y es siempre obligatorio (salvo en las referencias de tipo «`Misc`»). Designa el título del trabajo al que nos estamos refiriendo, salvo en las referencias tipo «`InBook`», donde designa el título del libro del que forma parte el capítulo o rango de páginas al que nos estamos refiriendo.

En teoría los estilos bibliográficos estándar *normalizan* (o pueden normalizar) el uso de las mayúsculas y minúsculas en los títulos. Y por ello en

---

de formatearlo correctamente. Incluso el paquete makebst es capaz de generar automáticamente un estilo bibliográfico que reconozca este campo. Véase la sección 4.1.

las referencias generadas a partir de nuestra base de datos no siempre se escribirán los títulos tal y como aparecen en la base de datos. Para preservar una letra en mayúsculas hay que encerrarla entre llaves. Las llaves pueden ponerse también alrededor de una o varias palabras. Esto es especialmente importante si en el título se usan comandos de LaTeX, ya que en ellos una alteración de las mayúsculas y minúsculas se traduciría en un error.

Type: Este campo existe en «`InBook`», «`InCollection`»[28], «`MasterThesis`», «`PhdThesis`» y «`TechReport`». En los tres últimos tipos mencionados los estilos estándar añaden a la referencia, como texto aclaratorio, el nombre del registro. Por ejemplo: en una tesis doctoral, se escribe: "*Tesis doctoral*". Si en el campo «`type`» hay algún texto, se usará este texto en lugar del predeterminado. En los registros «`InBook`», aunque por defecto no se escribe ningún texto, sí se escribirá lo que se diga en este campo. Por lo tanto aquí puede escribirse una pequeña aclaración respecto del tipo de trabajo de que se trata, como: "Conferencia", "Disertación", etc.

Volume: En artículos de revista se usa para indicar en qué tomo de la revista se ha publicado dicho artículo. En libros que formen parte de una serie, para indicar el tomo o volumen, dentro de la serie en cuestión. En este último caso si se da algún valor al campo «`volume`» los estilos estándar ignorarán el contenido del campo «`number`».

Year: El año de la edición del libro o de publicación del artículo. Es otro de los datos presentes (de forma obligatoria u opcional) en todos los tipos de registros. Hay incluso estilos bibliográficos en donde las referencias se ordenan por el valor de este campo, y las citas en el texto lo incluyen también.

2.2.4. Algunos consejos sobre tipos de registros y campos

Cuando empezamos a manejar BibTeX la gran cantidad de tipos de registros existentes (catorce en total) puede abrumar. A continuación expongo algunos consejos, dictados por la experiencia, sobre cómo debemos proceder.

1. Hay tres tipos de referencia básica: artículos de revista («`article`»), libros completos («`book`») y partes de libro. Para las partes de libro a su vez hay tres posibilidades: Si la parte se identifica por un número de sección o de páginas, la referencia correcta es «`InBook`», si se identifica por un título, yo distingo, a su vez, según cada parte del libro tenga un autor distinto, en cuyo caso utilizo «`InProceedings`», o todo el libro pertenezca al mismo autor, en cuyo caso prefiero usar «`InCollection`». Ello es porque no me tomo

[28] La documentación oficial de BibTeX y los textos que he leído coinciden en que en los registros «`InCollection`» el campo «`type`» es opcional. Las pruebas que yo he hecho me llevan a pensar que en estos registros este campo es ignorado.

demasiado en serio el nombre del tipo de referencia, y atiendo más bien al contenido de los campos[29].

2. Tampoco hay que tomarse demasiado en serio el nombre de los campos. En ocasiones puede ser conveniente, por ejemplo, incluir la ciudad de edición junto con el campo reservado a la editorial, y en otros casos puede ser preferible usar para ello el campo «`address`». Lo importante es que seamos consistentes en nuestras decisiones.

3. La distinción entre campos obligatorios y opcionales es importante sólo para confirmarnos que hemos elegido el tipo de registro adecuado. Pero tampoco es una tragedia que en un caso concreto, algún campo obligatorio se quede sin rellenar.

4. Para la información que consideremos relevante pero para la que no haya ningún campo que la haya previsto, podemos usar el campo «`note`». Pero también podemos usar campos específicos creados por nosotros. En tal caso los estilos estándar de BIBTEX la ignorarán, pero podremos crear un estilo bibliográfico propio que la tenga en cuenta. Yo, por ejemplo, utilizo muchas traducciones, y estimo que el nombre del traductor es un dato importante para la referencia. Empecé usando para ello el campo «`note`», pero al final diseñé un campo opcional llamado «`traductor`». Sobre la escritura de estilos bibliográficos propios, véase la última parte de este documento. Aunque cuando decidamos crear un campo adicional, si el tipo de datos que queremos incluir en él es relativamente corriente, puede merecer la pena que antes de bautizar el campo, nos demos un paseo por Internet; porque es muy posible que alguien haya ya sentido la necesidad de recoger tales datos, haya creado ese campo y, lo que es más importante, haya escrito un estilo bibliográfico que sepa qué hacer con él.

5. También es interesante la creación de campos adicionales para información que es importante para nosotros tener recopilada, pero que no importa a los lectores de nuestros documentos y no tiene por qué incluirse en las referencias bibliográficas generadas. Por ejemplo yo uso para los libros un campo llamado «`tejuelo`» en el que almaceno la "etiqueta" o "tejuelo" asignada a determinado libro en la biblioteca de mi centro de trabajo, lo que me ayuda a localizar físicamente el libro con rapidez.

6. Por último, aunque he señalado que no hay que tomarse demasiado en serio ni los nombres de los tipos de registro, ni los de los campos, ni su naturaleza obligatoria u opcional, y he añadido que puede ser interesante crear campos propios, todo ello no debe llevarnos a diseñar y escribir una base de datos que sólo nosotros seamos capaces de entender y utilizar. El sentido último

[29] Y no hay que escandilarse por ello. El propio autor de BIBTEX recomienda en [10] no tomarse excesivamente en serio los nombres de registros y campos.

de las herramientas de LaTeX (y en general del software libre) es el de *compartir* el trabajo: Que lo que ya ha sido hecho no tenga por qué repetirse (si está bien hecho y el autor autoriza su uso por otros). Cuando empezamos a escribir nuestra base de datos no hay forma de saber si terminará siendo tan importante que merezca la pena compartirla con otros y por ello es mejor empezar a escribirla con la mente puesta en que tal vez en el futuro debamos compartirla.

2.3. Otras cuestiones relativas a los ficheros «.bib»

2.3.1. Uso de abreviaturas y concatenación de cadenas

En un fichero «.bib» es posible definir abreviaturas que simplifiquen la escritura de los registros y eviten errores. Para usar una abreviatura, primero hay que definirla de acuerdo con el siguiente formato:

```
@String{Abreviatura = "Texto sin abreviar"}
```

El nombre de la abreviatura debe empezar por una letra y no puede contener caracteres en blanco ni ninguno de los siguientes caracteres:

```
" # % ' ( ) , = { }
```

El texto sin abreviar se debe especificar entre comillas. Las llaves para delimitar el contenido de "`@String`" son obligatorias.

La definición se puede hacer en cualquier lugar del fichero «.bib» que no sea el interior de un registro (o de algún otro elemento admitido en este tipo de ficheros). El único requisito en este punto es el de que la definición de una abreviatura debe hacerse *antes* de que ésta sea usada en algún campo.

Las abreviaturas no pueden usarse para los nombres de campo o de registro, sino exclusivamente en el contenido del campo. Y como las abreviaturas no pueden estar encerradas dentro de los delimitadores de campo, cuando todo el contenido del campo es una abreviatura, ese campo no requerirá delimitadores de contenido. Así en el siguiente ejemplo:

```
@STRING{TeX = "The {T}e{X} {B}ook"}

@Book{eijkhout,
   author = {Victor Eijkhout},
   title = {TeX by topic},
   publisher = {Addison Wesley},
   year = 1992,
}

@Book{knuth,
   author = {Donald E. Knuth},
   title = TeX,
   publisher = {Addison Wesley},
```

```
   year = 1986,
}
```

Hemos definido la abreviatura “TeX” y dos registros, en el título del primero de ellos aparece la cadena “TeX”, pero BIBTEX no la toma como una abreviatura, porque tal cadena se encuentra dentro de unas llaves que delimitan el contenido de un campo. Por el contrario, en el segundo registro, el texto “TeX” usado como contenido del campo título, es tomado como un uso de la abreviatura, dado que no está dentro de ningún tipo de delimitadores de contenido de campo[30].

Si la abreviatura no cubre todo el contenido del campo, entonces debemos escribir entre delimitadores normales el contenido no cubierto por la abreviatura, y *concatenar* dicho contenido con el de la abreviatura.

En los ficheros «`.bib`» el carácter para concatenar cadenas de texto es “#”. Este carácter permite concatenar abreviaturas entre sí, cadenas de texto entre sí y abreviaturas con cadenas de texto. Por ejemplo, si en el fichero «`.bib`» de nuestro anterior ejemplo tuviéramos un libro llamado “*Comentarios a ‘The TeX Book’ de Knuth*”, y quisiéramos aprovechar la abreviatura que hemos definido para “*The TeX Book*”, podríamos escribir el campo «`title`» de dicho libro de la siguiente manera:

```
title = {Comentarios a ‘} # TeX # {’ de Knuth},
```

Hay tipos de datos, como el nombre de las revistas científicas, para los que es muy corriente el uso de abreviaturas. De hecho en ciertos campos de conocimiento circulan ficheros en el que se recogen las abreviaturas corrientemente usadas para las revistas más importantes. En tales casos podemos usar dichos ficheros simplemente indicando su nombre en la orden “`\bibliography`” antes del nombre de nuestra base de datos bibliográfica, y separando ambos nombre por una coma.

La siguiente línea, por ejemplo, provocaría que “`MiBiblio.bib`” (que cabe suponer que es nuestra base de datos) se concatenara con el fichero distribuido por la *American Mathematical Society* que contiene las abreviaturas habitualmente usadas para las revistas matemáticas:

```
\bibliography{mrabbrev,MiBiblio}
```

En los estilos estándar de BIBTEX hay ya varias abreviaturas definidas por defecto: Están definidos los nombres de los meses del año en inglés (véase las abreviaturas que mencioné al hablar del campo «`month`» en la nota 26), así como el nombre de varias revistas dedicadas al campo de la informática[31].

---

[30] En el ejemplo he escrito para la abreviatura “TeX”, pero, como BIBTEX no distingue entre mayúsculas y minúsculas, habríamos obtenido el mismo resultado escribiendo “TEX” o “tex” tanto en la definición de la abreviatura como más tarde cuando ésta es usada. Podríamos también haber definido “TEX” y luego usar “tex” o “TeX”.

[31] Las abreviaturas definidas en el estilo plain son las siguientes: acmcs (ACM Computing Surveys), acta (Acta Informatica), cacm (Communications of the ACM), ibmjrd (IBM Journal of Research and Development), ibmsj (IBM Systems Journal), ieeese (IEEE Transactions on Software

2.3.2. Referencias cruzadas entre registros

En los ficheros «`.bib`» se admiten dos tipos de referencias cruzadas:

1. Usando el comando de LaTeX "`\cite`".
2. Mediante el campo «`crossref`».

En cuanto al uso de "`\cite`", podemos usarlo dentro de un campo para hacer referencia a otro registro. Normalmente esto se hace en el campo «`note`», pero se puede hacer en cualquier otro. Esto provocará que al citar en nuestro documento LaTeX una referencia que a su vez cita a otra, ambas se consideren citadas en el documento, y se incluyan las dos en la lista de referencias bibliográficas[32].

Pero también es posible usar referencias cruzadas de otra manera: para que BibTeX sepa que el valor de ciertos campos que normalmente forman parte del contenido de un determinado tipo de registro, pero que en un registro concreto no han sido incluidos, debe ser tomado de otro registro distinto. Por ejemplo, si tenemos un libro homenaje a cierto autor, en el que varios autores han colaborado, cada uno de ellos con un artículo. Si quisiéramos volcar todos estos artículos a nuestra base de datos, podríamos:

1. Crear tantos registros como artículos haya e incluir en cada uno de ellos todos los datos requeridos, gran parte de los cuales serán idénticos en todos los registros.
2. Abreviar mediante referencias cruzadas, para los datos idénticos escribirlos una sola vez.

Para seguir el segundo procedimiento, deberíamos empezar por crear un registro del tipo "Proceedings" para el libro homenaje en sí mismo considerado, rellenar en él el valor de todos los campos que podamos, y luego en el registro correspondiente a cada uno de los artículos extraidos de dicho libro (que serían referencias

---

Engineering), ieeetc (IEEE Transactions on Computers), ieeetcad (IEEE Transactions on Computer-Aided Design of Integrated Circuits), ipl (Information Processing Letters), jacm (Journal of the ACM), jcss (Journal of Computer and System Sciences), scp (Science of Computer Programming), sicomp (SIAM Journal on Computing), tocs (ACM Transactions on Computer Systems), tods (ACM Transactions on Database Systems), tog (ACM Transactions on Graphics), toms (ACM Transactions on Mathematical Software), toois (ACM Transactions on Office Information Systems), toplas (ACM Transactions on Programming Languages and Systems), tcs (Theoretical Computer Science)

[32] Este es uno de los casos en los que BibTeX requerirá una compilación adicional. Porque el comando "`\cite`" incorporado a un registro bibliográfico no será compilado por LaTeX hasta que se compila una versión que incorpore el contenido del fichero «`.bbl`». E incluso es posible que al incorporar esta nueva cita, sea preciso, tras la segunda ejecución de LaTeX, ejecutar de nuevo BibTeX.

de tipo «InProceedings»), usar como valor para el campo «crossref», el de la clave que hayamos asignado al registro «Proceedings»[33].

El efecto de todo ello será que:

1. En los registros «InProceedings» que escribamos para cada uno de los artículos contenidos en el libro, podremos no incluir los campos que ya están en el registro correspondiente al libro general, pues al apuntar «Crossref» al registro correcto, BIBTEX sabrá que los datos que faltan en el primer registro deben ser tomados del segundo. Esto, además de reducir el tamaño de la base de datos, y ahorrar tiempo para la introducción de los mismos, contribuye a garantizar que todas las veces que estos datos aparezcan, estén escritos exactamente igual.

2. Si en un documento LATEX citamos al menos dos artículos que apunten a un mismo registro, automáticamente se incluirá en la lista final de referencias bibliográficas, además de los trabajos directamente citados por nosotros, el registro correspondiente al libro que los contiene.

Para que lo anterior funcione correctamente, es preciso que el registro *referenciado* mediante el campo «crossref» (el registro «Proceedings») se encuentre en la base de datos *después* de los registros referenciadores (los registros «InProceedings»).

Los estilos estándar de BIBTEX admiten referencias cruzadas en los siguientes casos:

1. Los registros de tipo «InProceedings» pueden hacer referencia a registros «Proceedings» o «Book».

2. Los registros de tipo «InCollection» pueden hacer referencia a registros «Book».

2.4. Herramientas para manejar ficheros «.bib»

Existen varias herramientas para la creación y mantenimiento de bases de datos de BIBTEX cuya utilidad fundamental es la de asegurarse de que todos los registros están correctamente escritos y de que no hemos cometido errores ni en el nombre del tipo del registro ni en el de los campos.

Como yo trabajo exclusivamente en sistemas Linux, en este apartado me concentraré en herramientas que funcionen en tales sistemas. Para sistemas MS-DOS, Mac-OS o Microsoft Windows existen otras herramientas que yo no he probado y

[33] De hecho, si lo pensamos bien, el único sentido que tiene la presencia del campo «booktitle» en los registros de tipo «Proceedings» es el de que este campo se use para llenar el campo del mismo nombre en los registros «InProceedings» que apunten a este.

sobre las que, en consecuencia, no puedo opinar. Existen asimismo herramientas escritas en java que, en teoría, funcionan en cualquier plataforma, aunque yo no las he probado. He leído buenos comentarios sobre dos de ellas: "jbibtexmanager" y "jabref", ambas son fáciles de localizar en Internet.

2.4.1. Herramientas gráficas

Existen varias herramientas gráficas que permiten trabajar con ficheros «`.bib`» mediante ventanas, botones, barras de herramientas, etc. Las utilidades ofrecidas por estas herramientas son bastante similares. Entre ellas he probado (porque están disponibles directamente en Debian):

pybliographer: Posiblemente sea la más completa de todas las que conozco (aunque admito que no es la que más me gusta, pues es muy difícil de manejar sin el ratón, y yo tengo alergia al ratón). Frente a otras herramientas ofrece las siguientes ventajas: que funciona también en modo de consola, que además del formato BIBTEX puede manejar otros formatos de bases de datos bibliográficas y, lo que es más importante: es capaz de convertir bases de datos desde unos formatos a otros.

Gbib: Es mi herramienta favorita en este grupo[34]. Forma parte del proyecto Gnome, aunque no suele instalarse por defecto. Posiblemente su principal inconveniente esté en el hecho de que no permite controlar el tipo de codificación en el que hay que grabar el fichero, sino que éste es guardado automáticamente en la codificación preestablecida para el sistema[35], aunque en realidad este defecto es predicable de las tres herramientas que recojo en la presente sección.

Tkbibtex: Una herramienta muy parecida a las anteriores: permite crear bases de datos nuevas, abrir bases existentes, añadir registros, hacer búsquedas, etc. Apenas le he echado un vistazo.

---

[34] Lo que no es mucho decir, pues yo soy usuario de GNU Emacs (del que me ocupo en la próxima sección).

[35] Esto puede ocasionar una tediosa labor de *reparación*. Sobre todo si se tiene en cuenta que hasta hace poco la codificación estándar para sistemas Linux españoles era «`latin1`» o «`latin15`» y en los últimos tiempos las distribuciones empiezan a migrar a «`utf8`». De modo que si en un sistema instalado recientemente abrimos una base de datos escrita hace algún tiempo, introducimos cambios en ella y los guardamos, cabe la posibilidad de que, si no tenemos control sobre las codificaciones usadas, todos los caracteres no anglosajones del fichero se transformen en caracteres ininteligibles que luego habrá que restuarar a mano.

2.4.2. El modo BIBTEX de GNU Emacs

Dado que los ficheros «`.bib`» son ficheros de texto, otra posibilidad es editarlos con algún editor de texto que esté provisto de alguna extensión que facilite de alguna manera el trabajo con ellos.

Desde este punto de vista casi todos los editores de texto "avanzados" son capaces de reconocer la sintaxis de un fichero «`.bib`», y algunos especialmente preparados para trabajar con el formato LATEX, como por ejemplo kile, incluyen incluso opciones de menú que permiten insertar registros bibliográficos completos, ahorrándonos el tener que recordar de memoria los campos obligatorios y opcionales de cada uno de los registros.

No obstante en este grupo mi favorito sin ningún género de dudas es GNU Emacs. De hecho mi afición a dicho editor se puede comprobar en el hecho de que he escrito y puesto en circulación por Internet dos documentos dirigidos a facilitar su aprendizaje: [2] y [1][36].

Sería demasiado largo recoger aquí todo lo que ya he dicho en [1] (donde dedico al trabajo con ficheros «`.bib`» casi diez páginas) , y por lo tanto me remito a dicho documento.

2.4.3. Otras herramientas disponibles en sistemas linux

Además de las indicadas existen muchísimas otras herramientas disponibles. Si nos centramos, por ejemplo, en una instalación estándar de Debian, las herramientas que se incluyen para trabajar con ficheros BIBTEX son las siguientes:

bib2bib: Utilidad de consola que lee uno o varios ficheros bibliográficos y filtra sus entradas con respecto a un determinado criterio, generando un nuevo fichero bibliográfico que contenga exclusivamente los registros filtrados.

bibclean: Utilidad de consola para imprimir en pantalla y comprobar la sintaxis de una base de datos BIBTEX. También es muy útil para reformatear todos los registros, unificando su aspecto visual.

bibcursed: Es un editor de ficheros «`.bib`» que funciona en modo de consola (sin necesidad de tener cargado el sistema gráfico). Puede ser útil para los poco amantes de las ventanas.

bibindex: Utilidad de consola que permite crear un índice binario que acelere las búsquedas hechas en una base de datos con biblook.

[36] [2] constituye una introducción general a GNU Emacs, para quien se aproxime por primera vez a este potente editor, y [1] contiene una explicación completa de las utilidades de GNU Emacs en relación con el formato LATEX, dentro de la cual se incluyen las facilidades para trabajar con ficheros de BIBTEX.

biblook: Utilidad de consola que permite realizar búsquedas rápidas en un fichero bibliográfico. Las búsquedas pueden además acelerarse mucho si se genera un índice binario para el fichero mediante la utilidad bibindex.

bibtex2html: Utilidad de consola que permite convertir una base de datos bibliográfica al formato HTML

bibtool: Utilidad de consola que permite realizar varias acciones sobre un fichero «`.bib`». Es una herramienta muy útil que incluye un muy completo manual de referencia.

Junto con estas herramientas, que he destacado exclusivamente porque están disponibles de modo directo para cualquier usuario de sistemas Debian (o distribuciones basadas en Debian, como Ubuntu), por Internet circulan numerosas utilidades o scripts para manipular bases de datos bibliográficas en formato BIBTEX, y en la CTAN están disponibles varias de ellas: no todas trabajan en todos los sistemas, y no todas son igual de útiles, pero merece la pena echarles un vistazo, por lo que recomiendo la consulta del *catálogo general* de la CTAN, organizado por materias, en «`http://texcatalogue.sarovar.org/bytopic.html`».

3. Los estilos bibliográficos

3.1. Descripción de los estilos estándar de BIBTEX

BIBTEX se acompaña de cuatro estilos llamados «`plain`», «`abbrv`», «`alpha`» y «`unsrt`». El estilo esencial es el primero, y los otros funcionan como él en todo salvo en uno o dos puntos concretos.

A continuación describiré estos estilos atendiendo exclusivamente a sus características principales y sin entrar en excesivos detalles (cuya descripción con palabras, por otra parte, ocuparía demasiado espacio). Mi consejo para ver exactamente las características de cada uno de los estilos es escribir un fichero «`.bib`» de prueba, incluir un registro de cada uno de los tipos admitidos, llenar *todos* los campos posibles de cada registro, e incluir dicha base de datos en un documento de prueba, en el que hayamos incluido un comando "`\nocite{*}`".

En la próxima explicación, menciono varias veces la noción *autoría*. Con ella me quiero referir a lo que ya se explicó en la página 16.

Estilo "plain". Sus características son:

1. La lista bibliográfica final se ordena alfabéticamente atendiendo a la autoría, y si hubiera más de una obra del mismo autor, se toma en cuenta al año de las mismas y después el título. Si sigue habiendo igualdad tras aplicar los criterios anteriores, el último criterio es el del orden en

el que fueron citadas y, para obras citadas simultáneamente mediante "`\nocite{*}`", el orden que tengan en la base de datos.

2. Las obras incluidas en la lista son numeradas consecutivamente y el número asignado a cada una de ellas, entre corchetes, se convierte en la etiqueta identificativa de la misma que será impresa en el lugar en el que se encuentren los comandos "`\cite`" existentes en el cuerpo del documento.
3. Los datos de los campos se incluyen completos.
4. Para ciertos campos se añaden determinadas palabras o abreviaturas en inglés. Por ejemplo, tras el contenido del campo «`edition`» se añade la palabra "*edition*", y hay tipos de registro en los que el nombre del campo (en inglés) forma parte de la referencia.

El estilo "abbrv": Es idéntico al estilo «`plain`» salvo en el hecho de que para ciertos datos se usan abreviaturas y así el nombre de pila de los autores es sustituido por sus iniciales y el nombre de ciertas revistas (que están predefinidas en el estilo y que se refieren a la informática) es sustituido por su abreviatura. Para la mayor parte de los usuarios, que no usan las revistas predefinidas en el estilo, la única diferencia con «`plain`» es que del nombre propio del autor sólo se usan las iniciales.

El estilo "alpha": Se distingue del estilo «`plain`» exclusivamente en el hecho de que la etiqueta de identificación de cada obra en la lista no es un número, sino un texto generado automáticamente a partir de la *autoría* de la referencia, el año de publicación y, en ocasiones, el inicio del título[37]. Asimismo la lista bibliográfica se ordena alfabéticamente según las etiquetas asignadas, y el comando "`\cite`" imprime, en el lugar en el que se encuentre, la etiqueta asignada a la obra citada.

El estilo "unsrt": Es igual al estilo «`plain`» salvo en el hecho de que en él la lista bibliográfica no se ordena alfabéticamente, sino según el orden en el que las distintas obras que aparecen en ella fueron citadas por primera vez. En este caso, para las obras incluidas en la lista de referencias mediante un comando "`\nocite`" se considerará que fueron citadas en el lugar en el que se encuentre tal comando. Y para el comando "`\nocite{*}`", se usará el orden en el que las referencias se encuentren en el fichero «`.bib`», pero sólo para las referencias que no hubieran sido citadas antes de "`\nocite{*}`".

[37] Como regla se toman las tres primeras letras del nombre del autor y las dos últimas cifras del año de publicación. En obras con varios autores se toma la inicial de los tres primeros seguida de un signo "+" en formato de superíndice, si hay más de tres autores. Si con estos criterios a dos referencias se les asignara la misma etiqueta, las obras se ordenan relativamente entre sí, atendiendo a los criterios de ordenación del estilo «`plain`», y a las etiquetas se añade una letra para diferenciarlas entre sí: una "a" a la primera, una "b" a la segunda, etc.

En cuanto a las características concretas de formateo de los campos son similares en todos estos estilos. Los aspectos más llamativos son:

1. Se escribe el nombre de pila antes que el apellido, lo que no deja de ser chocante en aquellos casos en los que la lista de referencias se ordena alfabéticamente a partir de los apellidos (estilos «`plain`» y «`abbrv`», ya que ello provoca una ordenación alfabética tomando como referencia, no la primera palabra de cada párrafo, sino la segunda o la tercera.

2. Los distintos elementos de la referencia se separan mediante un punto. Lo que también provoca problemas cuando el contenido de un campo no es numérico y no empieza con una mayúscula, pues los estilos sólo ajustan el uso de las mayúsculas en el campo «`title`».

3. Todos los campos aparecen con letra normal, salvo el título de los libros y trabajos académicos y el nombre de las revistas, que aparecen en cursiva.

3.2. Estilos adicionales

Junto con estos estilos estándar, existen multitud de estilos disponibles. En la CTAN hay varios de ellos preparados para su descarga, y algunas distribuciones de LaTeX incluyen por defecto ciertos estilos adicionales[38].

Entre los estilos adicionales hay algunos cuya instalación es tan corriente que son estilos *casi* estándar. Otros no son tan conocidos, pero resultan interesantes. Muchos de ellos consisten exclusivamente en uno o varios estilos para BibTeX, y otros incluyen también un *paquete* que hay que incluir en el documento principal, mediante "`\usepackage`", para poder usar el estilo.

No voy a explicar todos los estilos adicionales, porque ello sería tedioso, sino exclusivamente los más conocidos e interesantes:

abstyles: Son un conjunto de estilos llamados "*estilos adaptables*". Su principal peculiaridad es que ciertas características del estilo están *parametrizadas* de tal manera que pueden ser cambiadas desde el documento «`.tex`» mediante ciertos comandos, aunque en este punto la información que ofrece su fichero de ayuda es verdaderamente críptica. Entre sus inconvenientes puede citarse el hecho de que sólo están preparados para trabajar con documentos en alemán o en inglés. Aunque esto tiene arreglo si en lugar de los «`abstyles`» propiamente dichos usamos "`abstyles-babel`" que es una modificación

[38] Por ello, antes de buscar estilos adicionales, conviene que nos aseguremos de cuáles son los estilos incluidos en nuestra distribución de LaTeX. Lo que se puede hacer por distintos procedimientos que dependen de nuestra concreta instalación. A falta de algún otro procedimiento específico, podemos buscar los ficheros de extensión «`.bst`» existentes en nuestro sistema; o mejor: los existentes en alguno de los directorios en los que BibTeX busca los ficheros de estilo. En la nota 7 se explica cómo localizar cuáles son esos directorios en un sistema Unix/Linux.

de «abstyles» para que si en babel el idioma activo es "spanish" el estilo bibliográfico se genere en español. Pero como los estilos que proporcionan ambos paquetes tienen el mismo nombre, si tenemos instalados los dos paquetes en nuestro sistema, sólo funcionará uno de ellos: el primero que sea localizado por BIBTEX cuando busque el fichero de estilo[39].

achicago: El estilo de citas basado en el "*Chicago Manual of Style*" está implementado por varios paquetes: «achicago», «achicago.bst», «chicago», «jas99», «jas99m» y «newapa». También el paquete «authordate» produce citas acordes con el manual de estilo Chicago, aunque este último paquete tiene otras peculiaridades.

El estilo de citas Chicago es del tipo Autor-año y, personalmente, para obtener citas de estas prefiero el paquete natbib (que se menciona en seguida).

Apalike: Este estilo también fue escrito por Oren Patashnik (el autor de BIBTEX). Es un estilo de tipo "Autor-año" y para su correcto funcionamiento, hay que cargar en los documentos LATEX el paquete del mismo nombre que el estilo: «apalike».

Harvard: La familia de estilos Harvard implica también citas del tipo Autor-año.

Natbib: Natbib es al mismo tiempo un estilo y un paquete, aunque quizás en él destaque más la parte de paquete, por la cantidad de modificaciones que incorpora al funcionamiento del comando "\cite".

Durante mucho tiempo (hasta que conocí el trabajo de unos compañeros de la Universidad de Murcia que han diseñado «flexbib»), he considerado que «natbib» era la mejor opción. Permite elegir entre citas del tipo Autor-año o citas numéricas. Incluye versiones adaptadas de los estilos estándar (salvo el estilo «alpha») llamadas «plainnat», «abbrvnat» y «unsrtnat»; añade un campo «usr», dispone de una estupenda documentación e incorpora numerosas modificaciones y mejoras en el procedimiento de las citas.

Flexbib: Este paquete ha sido diseñado en la Universidad de Murcia, y no está disponible en la CTAN, sino en «http://www.latex.um.es/». Incorpora muchas de las ventajas de «natbib» y a ellas añade dos muy importantes: La primera es que este paquete es sensible a «babel», la segunda es que muchas de las características esenciales del estilo están *parametrizadas*, es decir: pueden cambiarse sin necesidad de cambiar de estilo, porque funcionan como *opciones* del paquete. Una muy buena descripción del paquete, además de en su página web, en [9].

[39] Cuál sea el primero depende de la concreta distribución de LATEX que tengamos instalada, y de la ruta de búsqueda de ficheros que en ella se haya implementado. Cuando LATEX (o BIBTEX) busca un fichero, va mirando por orden en diferentes directorios hasta que lo encuentra.

3.3. Cómo españolizar los estilos estándar

Los estilos estándar de BIBTEX están pensados para documentos que se vayan a redactar en inglés y por lo tanto usan partículas, palabras y abreviaturas en inglés. Para conseguir que nuestra lista de referencias figure totalmente en español, sin palabras en inglés, existen varios procedimientos.

1. Lo más cómodo es usar algún estilo o paquete que añada a los paquetes estándar la sensibilidad a «`babel`». Hay varios paquetes que hacen eso, y más cosas. Podemos citar "`abstyles-babel`", diseñado por Tomás Bautista y que es una adaptación de los abstyles, para que sean sensibles a «`babel`», o los paquetes «`babelbib`» y «`flexbib`».

2. Usar algún estilo diseñado para el idioma español. En Internet se pueden localizar algunos, y en la CTAN está disponible el estilo "spain.bst" basado en las indicaciones que sobre referencias bibliográficas se contienen en el "Diccionario de ortografía técnica" de J. Martínez de Sousa.

3. Podemos también generar nuestro propio estilo bibliográfico, preparado para trabajar en un idioma concreto. Al respecto estúdiese lo dicho en la sección 4.

4. Por último, podemos modificar el fichero donde se contiene algún otro estilo, y prepararlo para que trabaje en español. Esta última posibilidad exige un mayor conocimiento del funcionamiento interno de BIBTEX. Conviene por lo tanto aprender muy bien lo que se expone en la parte III de este documento.

Para esta parte de la guía, referida al nivel básico, el procedimiento más adecuado es el primero de los mencionados.

Parte II
BIBTEX intermedio

4. Generación automatizada de estilos bibliográficos

Si ninguno de los estilos bibliográficos que acompañan a BIBTEX, o que están disponibles en la CTAN nos convence, podemos generar nuestro propio estilo bibliográfico. Ello se puede hacer escribiendo a mano nuestro propio estilo, lo que exige un profundo conocimiento del lenguaje usado por los ficheros de estilo (véase la parte III), o mediante algunas herramientas dirigidas a la generación de ficheros «`.bst`». Estas herramientas, cuando son ejecutadas, empiezan preguntando al usuario por las características que debe tener el nuevo estilo (preguntas en inglés, ¡of course!), y, con las respuestas dadas, se genera un fichero «`.bst`» que contiene el nuevo estilo o, lo que es más normal, un fichero intermedio a partir del cual es posible generar el fichero «`.bst`».

La herramienta que voy a analizar aquí funciona de esa segunda manera: el fichero intermedio tiene la extensión «`.dbj`» y la herramienta se llama «`makebst`» y forma parte del paquete «`custom-bib`» (disponible en la CTAN).

4.1. Makebst

Makebst es un programa para TEX diseñado por Patrick W. Daly que permite generar ficheros bibliográficos personalizados. Para instalarlo[40] hay que descargarlo desde la CTAN, descomprimirlo en algún directorio y ejecutar, sucesivamente, los siguientes comandos

```
$> latex makebst.dtx
$> latex makebst.ins
```

El primero de estos comandos extraerá del fichero «`makebst.dtx`» la documentación y el fichero «`makebst.ins`». El segundo comando genera un fichero llamado «`makebst.tex`» que será el que usaremos para generar nuestros estilos personalizados.

En el paquete de instalación, junto con «`makebst.dtx`» se encuentran varios ficheros de extensión «`.mbs`». El nombre MBS proviene de las siglas de «*Master* BIBTEX *Style*». Para la generación de nuestro estilo personalizado deberemos usar dos de esos ficheros: «`merlin.mbs`» que es el fichero maestro, y un segundo

[40] OJO: Hay distribuciones de LATEX que instalan por defecto este paquete, por lo que lo mejor es, antes de nada mirar en el directorio predeterminado para los paquetes incluidos con la distribución. En mi sistema ese es «`/usr/share/texmf-tetex/tex/latex/`». En el caso de que el paquete ya esté instalado, muy posiblemente no sean precisos los pasos que se indican en el texto.

fichero «`.mbs`» que contenga las especificaciones adecuadas al idioma en el que se generará el nuevo estilo. Para que el nuevo estilo esté españolizado ese fichero debe ser «`spanish.mbs`», pero podemos también generar estilos en otros idiomas: catalán, esperanto, portugués... etc. También podemos generar un estilo sensible a babel, aunque eso sólo funciona con algunos idiomas, entre los que afortunadamente se encuentra el español.

Para generar un estilo, simplemente debemos ejecutar

```
latex makebst.tex
```

Ello pondrá en marcha un proceso interactivo en el que se nos harán una serie de preguntas respecto del estilo a generar. Las preguntas van en inglés y, una vez que hemos empezado no es posible la marcha atrás, por lo que si nos equivocamos en alguna respuesta, debemos elegir entre interrumpir el proceso y volver a empezar desde el principio (lo que se hace pulsando primero CTRL-C y después X), o seguir adelante para más tarde modificar a mano el error. Al respecto véase la sección 4.2.

Explicar las distintas preguntas, por el orden en el que aparecen, es complejo, puesto que las preguntas planteadas por makebst son interactivas, es decir: dependiendo de la respuesta dada a cierta pregunta, se plantearán o no otras preguntas adicionales. Además no todas las preguntas tienen la misma importancia y las preguntas se plantean a veces de manera desordenada (en mi opinión), lo que dificulta su tratamiento sistemático. Por ello a continuación en lugar de referirme a todas y cada una de las preguntas que se plantean durante el proceso, las agruparé temáticamente por el orden en el que se plantean, explicando, para cada grupo, la utilidad general de las distintas preguntas, y resaltando las que me parecen más importantes.

Como no voy a explicar el significado de todas las preguntas, lo que si haré es dar un consejo general para el caso de que no sepamos que respuesta dar a alguna de ellas: Aceptar la respuesta predeterminada, es decir: pulsar ENTER para pasar a la próxima pregunta. Cuando se enumeran varias respuestas posibles, la respuesta por defecto viene marcada con un asterisco. En los demás casos la respuesta por defecto se indica entre paréntesis, tras la pregunta. La única pregunta que no admite una respuesta por defecto es la relativa al nombre del fichero de salida, es decir: el nombre que tendrá el estilo que queremos crear.

Las preguntas que makebst nos hace son las siguientes:

1. **Preguntas generales.** En el primer grupo de preguntas (las cinco primeras) se plantean una serie de cuestiones generales que afectan a ciertos aspectos del funcionamiento de makebst, y al fichero de estilo que se generará. De estas preguntas las más importantes son la 2ª y la 3ª que plantean las siguientes cuestiones:

    - Nombre del fichero maestro. El fichero maestro es el fichero con base en el cual makebst planteará las próximas preguntas y generará el fi-

chero de estilo solicitado. El valor por defecto es merlin.bst, y este es el valor que en el resto de esta sección asumo que hemos introducido. Si lo cambiamos, las preguntas que se plantearán serán distintas a las que aquí recojo[41].

- Nombre del fichero a generar. Es decir: el nombre que tendrá el estilo que estamos generando. Por ejemplo: "MiEstilo". El nombre que indiquemos tiene que ser correcto para nuestro sistema operativo. No hay que especificar la extensión que tendrá el fichero, ya que automáticamente se añadirá la extensión «`.bst`».

Para el resto de las cuestiones planteadas podemos dejar el valor por defecto.

2. **Preguntas relativas a los ficheros externos.** Makebst utiliza (o puede utilizar) dos ficheros externos, en conjunción con el fichero maestro (que por defecto es merlin.bst). Estos ficheros son uno para la definición de idioma, y otro con nombres extra de revistas:

   - Fichero para la definición de idioma. Este fichero es esencial porque de él depende el idioma en el que nuestro fichero de estilo generará las listas de bibliográficas. Con el paquete custom-bib se incluyen varios ficheros de definición de idioma (catalán, danés, holandés, inglés, esperanto, finés, francés, etc). Para el español hay que usar el fichero «`spanish.mbs`». Si aquí no se elige ningún fichero externo de definición de lenguaje, más tarde se nos preguntará si queremos generar un estilo en inglés (idioma por defecto) o un estilo sensible al idioma activo en el paquete babel. Esta sería la vía para generar un estilo multilingüe. Aunque para que este estilo multilingüe funcione, es preciso disponer de un fichero llamado «`babelbst.tex`»[42].

[41] Además, para cambiarlo, deberíamos tener algún otro fichero que cumpla su función. Yo no lo conozco, y quien sepa escribirlo, con toda seguridad que no necesita ejecutar makebst para crear su propio fichero de estilo.

[42] En teoría «`babel`» debería proporcionar un conjunto de ficheros cada uno de los cuales contuviera la traducción a un idioma de las cadenas de texto usadas por BibTeX, junto a un fichero general, «`babelbst.tex`» que se limitara a cargar el fichero correspondiente al idioma activo. Sin embargo babel no proporciona el fichero «`makebst.tex`» y además son muy pocos los idiomas para los que existe un fichero de definición de idioma para usar con BibTeX. Se supone que el nombre de esos ficheros consiste en el código del idioma correspondinte (por ejemplo: "es" para el español, "fr" para el francés, "it" para el italiano, etc) seguido de "bst.tex". O sea, el fichero para el español se debe llamar "esbst.tex". Y aunque hay pocos ficheros de definición de idioma, afortunadamente el español es uno de ellos. Es decir: el fichero «`esbst.tex`» existe. Normalmente se encontrará en el directorio en el que tengamos instalado el paquete babel.

Pero seguimos sin tener un fichero llamado «`babelbst.tex`», necesario para que el estilo creado con makebst funcione. Podemos crearlo de dos formas: Escribiéndolo nosotros (difícil si no se sabe el bastante TeX), o cambiándole el nombre a «`esbst.tex`» por «`babelbst.tex`» (o creando un enlace, o una copia). Si hacemos esto último, tendremos un estilo sensible a babel pero que sólo será capaz de manejar dos idiomas: el inglés y el español. No estoy seguro de que eso merezca el nombre de "estilo multilingüe", pero, afortunadamente, a los hispanoparlantes nos sirve.

- Ficheros extra con nombres de revista. Se trata de algunos ficheros incluidos en el paquete custom-bib en los que se recogen abreviaturas estándar para las revistas más importantes y conocidas de ciertos campos científicos[43]. Si incluimos estos ficheros haremos que el estilo que estamos generando incluya abreviaturas preestablecidas para las revistas incluidas en tales ficheros.

3. **Preguntas relativas al estilo de las citas:** Este grupo de preguntas posiblemente sea el que en mayor medida marque el tipo de estilo que estamos generando. Las citas pueden ser básicamente de los siguientes estilos:

   a) Estilo estándar LaTeX. Es lo que se llama *estilo numérico*: a cada obra incluida en la lista bibliográfica final se le asigna un número, el cual es el que se imprime en los lugares del texto donde dicha obra sea citada mediante el comando "`\cite`". Este es el estilo de citas que usan tres de los cuatro estilos estándar de BibTeX: «`plain`», «`abbrv`» y «`unsrt`». También es el estilo que usa LaTeX cuando la bibliografía se construye sin la ayuda de BibTeX. Este estilo es el estilo por defecto para makebst.

   b) Estilo Autor-año. Este estilo es un estándar en determinados campos científicos y admite diferentes modalidades (natbib, apalike, harvard, astronomy, chicago). Básicamente consiste en que para cada obra incluida en la lista de referencias se genera una etiqueta que contiene el apellido completo del autor y el año de publicación. Las distintas modalidades inciden en cómo se separan ambos elementos, y afectan también a qué hacer en el caso de que haya varios autores. Si el mismo autor tiene varias obras el mismo año, se diferencian unas de otras añadiendo una letra al año, como, por ejemplo, en «`Ataz 1980a`».

   c) Estilo «`alpha`». En cierto modo se puede considerar una modalidad del estilo Autor-año, porque para cada obra incluida se genera una etiqueta a partir del apellido del autor, año de la obra y, en su caso, inicio del título. Pero la diferencia está en que mientras el estilo Autor-año pretende que el lector sólo con la lectura de la etiqueta se haga una idea aproximada de qué obra se ha citado, en el estilo «`alpha`», la etiqueta no pretende tener sentido, sino que simplemente sirve para identificar a cada obra citada en el trabajo de manera única. Por ello normalmente no se usa el apellido completo, ni este se separa del año, el cual tampoco suele ponerse completo. Esta es la razón de que cuando se habla de estilos tipo "Autor-año" el estilo «`alpha`» nunca se entienda incluido en tal expresión.

[43] Los ficheros son: «`geojour.mbs`»: Revistas de geografía física; «`photjour.mbs`»: Revistas de óptica; «`physjour.mbs`»: Revistas de física y «`supjour.mbs`»: Revistas suplementarias.

Para el estilo «alpha» makebst admite tres modalidades, en las que la etiqueta se construye de diferente manera. En la primera, cuando hay varios autores, la etiqueta se construye a partir de la inicial de cada autor. En la segunda modalidad, cuando hay varios autores, se usa el apellido del primero, como si hubiera un solo autor. En la tercera modalidad, el apellido del autor se incluye entero en la etiqueta.

d) Estilo en el que el contenido de la etiqueta identificativa de las obras en la lista de referencias es igual a la clave interna de la referencia en el fichero «.bib». Este estilo está pensado exclusivamente para cuando se quieran generar listados del contenido de una base de datos bibliográfica.

Pues bien: en la primera pregunta de este grupo de preguntas deberemos elegir entre uno de estos seis estilos (cuatro estilos básicos de los que uno de ellos admite tres modalidades).

Dependiendo del estilo elegido, las siguientes preguntas varían. Y así:

a) Si se eligió estilo tipo Autor-Año, se nos preguntará, de las distintas variedades que este estilo tiene (natbib, apalike, harvard, astronomy, chicago) cuál preferimos, pudiendo incluso elegir una variedad innominada, siempre que dispongamos de un fichero «.sty» que la implemente.

b) Si no se eligió el estilo Autor-Año, se nos preguntará si queremos que se genere salida HTML. Se trata de una utilidad que permite que el fichero de estilo, en lugar de generar un fichero «.bbl» construido con comandos LaTeX, genere un fichero HTML, lo que sería una forma fácil de escribir una página HTML que incluya un listado bibliográfico. Aunque la verdad es que hay otras utilidades para eso que me gustan más. En todo caso, si decidimos contestar que sí, podemos elegir tres modalidades para el HTML. En la primera las referencias se separarán por párrafos; en la segunda, las referencias se incluirán en una lista HTML, y en la tercera se incluyen las claves internas de las referencias. Si aceptamos cualquiera de estas posibilidades hay que tener en cuenta que cualquier intento de compilar nuestro documento principal tras haber ejecutado BibTeX sobre él y generado el fichero «.bbl», generará un error, porque el tal fichero «.bbl» internamente no consta de comandos LaTeX, sino de comandos HTML.

4. **Preguntas relativas a campos adicionales:** Las siguientes dos preguntas afectan a los campos en nuestra base de datos. Si hemos construido un fichero «.bib» en el que sólo estén los campos estándar, hay que contestar a ambas que no. Pero es posible que hayamos añadido dos campos extra para

los registros, y contestando "y" (de "yes") a estas dos preguntas, conseguiríamos que tales campos sean reconocidos por el estilo. Los campos a que se refieren estas preguntas son los siguientes:

a) Campo «`language`». Si en nuestra base de datos las referencias tienen un campo llamado «`language`» cuyo contenido es el nombre del idioma en que está el título de la referencia en cuestión, tal y como hay que referirse a dicho idioma en «`babel`», y contestamos que si a esta pregunta, en nuestro estilo se activará el idioma correspondiente a cada referencia antes de incluirla en la lista de referencias. Eso garantiza que la partición en guiones del título sea correcta de acuerdo con las reglas correspondientes al idioma en el que dicho título esté escrito. En caso de que para alguna referencia no constara valor para este campo, se usaría el idioma por defecto para el documento. Téngase en cuenta, además, que para usar esta utilidad, en el encabezamiento de nuestro fichero «`.tex`», al incluir la orden "`\usepackage{babel}`" hay que especificar como idiomas a cargar todos aquellos que se encuentren representados en la base de datos.

b) Campo «`annote`». Este campo se puede haber incluido para anotaciones extra sobre la referencia a que se refiere un registro. Si disponemos de él, y contestamos que sí a la pregunta que nos haga makebst, su contenido será incluido en la lista de referencias final.

5. **Orden de las referencias:** La siguiente pregunta también es muy importante, aunque sólo se hace en el caso de que no hayamos elegido el estilo de citas «`alpha`». Se trata del orden en el que queremos que aparezcan las referencias citadas en la lista de referencias. La ordenación posible varía dependiendo del estilo de citas elegido. Si se eligió como cita el estilo numérico (o el basado en la clave), podemos elegir entre:

   - Orden alfabético de autores. Es la respuesta por defecto.
   - Orden de citas. Funciona del modo similar al estilo estándar «`unsrt`».
   - Orden cronológico (por el valor del campo «`year`») y después por autores.
   - Orden cronológico inverso, es decir: como el anterior, pero empezando por las referencias más recientes.

   Pero si elegimos el estilo de citas Autor-año, entonces la segunda opción es la de ordenar por etiquetas, sistema este que es exigido por algunas revistas y que provoca una ordenación más sensible que la alfabética pura por autores. Las diferencias están en el caso de que haya autores con obras firmadas por ellos solos y con obras en colaboración.

6. **Formateo de los autores y fechas**. A continuación se incluyen varias preguntas (12 en total, creo) relativas a cómo formatear el nombre de los autores y las fechas. Yo supongo que estos campos se tratan antes que los demás, porque son los que se usan en el estilo de citas Autor-año, y porque en la lista de referencias que se generará, el campo autor es siempre el primero.

   Las cuestiones aquí planteadas son muchas y, si quiero evitar que este documento se haga interminable, no puedo ocuparme de todas de forma pormenorizada. A título de ejemplo enunciaré las siguientes:

   - Cómo tratar las partículas que separan el nombre de los apellidos. Es decir: "Miguel de Cervantes" ¿debe clasificarse en la "D" como "De Cervantes, Miguel", o en la "C" como "Cervantes, Miguel De"? Recordemos, a efectos de esta pregunta, que BIBTEX identifica que una parte de un nombre es partícula que separa el nombre de pila de los apellidos en el caso de que dicha parte esté escrita con minúsculas.
   - Cómo escribir el nombre de los autores. Podemos elegir entre varias posibilidades que combinan varios factores, como si escribir o no el nombre de pila completo, si colocar o no los apellidos antes que el nombre, si poner o no un punto detrás de las iniciales, etc. En total se ofrecen diez posibilidades para esta pregunta.
   - Tipo de letra para el nombre de los autores: Normal, versalitas, negrita, etc.
   - Lugar en el que hay que imprimir la fecha.
   - Tipo de letra para la fecha.
   - Etc. La verdad es que las preguntas son exhaustivas y si me ocupara de todas este documento llegaría al centenar de páginas. Hay preguntas para, por ejemplo, la puntuación entre los nombres de los autores, qué hacer en el caso de que se repitan en la lista dos obras del mismo autor, qué hacer en el caso de que haya demasiados autores para una referencia, cómo debe aparecer el nombre del autor en las etiquetas (si se eligió un estilo de citas tipo Autor-año), qué hacer cuando en algún registro no conste la fecha, qué uso dar al contenido del campo «`month`», cómo ordenar los datos que componen la fecha, etc.

7. **Formateo de los distintos campos**: El siguiente grupo de cuestiones (más de 30) afecta al formateo que recibirán los distintos campos en los diferentes tipos de registro, aunque también se cuelan algunas preguntas extra sobre ciertos campos adicionales a los estándar. Este grupo de preguntas es el que me parece especialmente desordenado, o en el que yo no he sabido descubrir cuál es el orden. Se empieza preguntando por el campo título de los artículos, se sigue preguntando por otros datos sobre artículos y, de pronto, se salta al título en las tesis doctorales, el título y número en los informes técnicos, para

volver otra vez a las revistas... Las preguntas son numerosas, pero no muy difíciles de entender si se tienen presentes los tipos de registro y los campos de cada uno de ellos.

8. **Preguntas relativas a la puntuación entre las distintas partes de la referencia bibliográfica**. En este grupo de preguntas (en total cinco) podemos elegir el tipo de puntuación que, de modo general, separará a unas partes de otras en la referencia bibliográfica (punto, coma, punto y coma) y el tipo de puntuación que se usará para separar ciertos campos específicos. Estos campos son:

   - El campo «`author`». Es el primer dato de las referencias, y muchas personas tienen la costumbre de separar este dato del resto mediante el signo de los dos puntos, cuyo uso no tiene sentido para separar el resto de los campos.
   - El campo «`note`». Suele ser el último dato y podemos elegir para él que siempre vaya precedido de un punto, aunque el resto de los campos se separen entre sí de otra manera.

9. **Opciones dependientes del idioma:** Si en la pregunta correspondiente elegimos un fichero de definición de idioma, a partir de aquí las preguntas serán extraidas de dicho fichero. En total las preguntas son 9, y en ellas se nos plantea si se deben usar o no ciertas palabras y abreviaturas tales como "páginas", "editor", etc. Ello implica en ocasiones volver a temas ya tratados como, por ejemplo, el formateo de los nombres de los autores, ahora para ver cómo usar la partícula separadora de dos o más autores, la cual varía dependiendo del idioma.

   Ese volver a temas anteriores, contribuye a que, como antes dije, se produzca la sensación de que las preguntas se plantean de manera desordenada.

10. **Otras preguntas:** Las siguientes preguntas (cuatro o cinco), acentúan aun más la sensación de desorden de la que hablaba, porque se vuelve otra vez a preguntar sobre posibles campos adicionales en nuestra base de datos. En tal sentido se nos pregunta si usamos o no campos adicionales para REVTeX[44]. Entre estas preguntas es especialmente importante, en mi opinión, la relativa a si usamos o no un campo «`url`». Si en nuestra base de datos hemos añadido, para documentos electrónicos, un campo url destinado a almacenar su dirección en Internet, constestamos que sí en esta pregunta el contenido de este campo se incluirá en la referencia bibliográfica y se formateará correctamente, aunque, para esto último, es preciso que en nuestro documento LaTeX hayamos cargado el paquete «`url`».

[44] El paquete REVTeX, de la Sociedad Física Americana (APS) da soporte para algunos campos adicionales en las bases de datos: collaboration, eid, eprint, archive, numpages y url.

11. **Preguntas relativas a la compatibilidad de los comandos a utilizar en el fichero «`.bbl`»:** Las dos penúltimas preguntas se refieren a qué comandos usar dentro del fichero «`.bbl`» que es generado por BIBTEX. Ello afecta a la compatibilidad de dicho fichero con las distintas versiones de LATEX. En concreto las preguntas se refieren a cómo generar la cursiva, y a la compatibilidad o no con plain TEX.

12. **La última pregunta:** Finalmente se nos informa que se ha generado un fichero «`.dbj`» y se nos pregunta si queremos ejecutarlo. Si respondemos que sí se generará ya nuestro fichero de estilo. Si respondemos que no (respuesta por defecto) para generar el fichero de estilo deberemos ejecutar nosotros mismos el fichero «`.dbj`», lo que se explica en la próxima sección.

4.2. Uso de los ficheros «`.dbj`»

Al terminar la ejecución de makebst, en el directorio desde donde se le hubiera llamado un fichero con el nombre que hayamos indicado al principio del proceso (en la tercera pregunta) y con extensión «`.dbj`». Si a la última pregunta respondimos que sí, tendremos además un fichero, del mismo nombre y con extensión «`.bst`». Este último fichero es nuestro fichero de estilo, y por lo tanto para usarlo basta con copiarloa alguno de los directorios en los que BIBTEX busca los ficheros de estilo[45].

Pero de momento es mejor concentrarnos en el fichero «`.dbj`». En él se contiene la información necesaria para generar un fichero de estilo con las características que hemos especificado a lo largo del proceso que se acaba de ver. El fichero de estilo se generará simplemente ejecutando LATEX sobre el fichero «`.dbj`». Así, por ejemplo, si nuestro estilo se llama «`MiEstilo`», para generar el fichero «`MiEstilo.bst`» nos basta con, en la línea de comandos, ejecutar

```
latex MiEstilo.dbj
```

Por lo tanto, si queremos generar un nuevo fichero de estilo, muy parecido al que acabamos de generar, pero con alguna variante, es mucho más cómodo *modificar* el fichero «`.dbj`» que volver a ejecutar makebst. Asimismo, si durante la ejecución de makebst, en algún momento nos equivocamos, en lugar de interrumpir el proceso y volver a empezar, es más razonable terminarlo y luego modificar el fichero «`.dbj`».

En el fichero «`.dbj`» se almacenan todas las preguntas planteadas por makebst, a partir de las relativas al uso de ficheros externos, con sus correspondientes respuestas. Tanto las preguntas como las respuestas posibles van, en general, precedidas de un signo de comentario (`%`). Y para cada pregunta se distingue entre la respuesta por defecto y las restantes. La respuesta por defecto aparece sangrada

[45] Lo que depende de nuestra concreta distribución. En la distribución estándar para sistemas Unix/Linux, ese directorio sería «`/usr/local/share/texmf/bibtex/`».

con respecto a las demás, y en las restantes respuestas, tras el signo % al principio de la línea, hay una cadena de texto, seguida de una coma y un nuevo signo % tras el que se contiene el texto de la respuesta.

Pues bien: para cada pregunta hecha por makebst y recogida en el fichero «`.dbj`»:

1. Si todas las respuestas empiezan con una marca de comentario, significa que la respuesta a esa pregunta coincide con el valor por defecto (siempre la primera respuesta, algo más sangrada que las restantes).

2. Si alguna línea no empieza por una marca de comentario, significa que esa fue la respuesta elegida.

Por lo tanto, para cambiar algún detalle en un estilo, disponiendo del fichero «`.dbj`» es más sencillo modificar a mano este fichero que volver a ejecutar makebst. Tras modificar el fichero «`.dbj`», ejecutando LaTeX sobre él se generará un nuevo fichero de estilo con la nueva característica activada.

Por ejemplo: Imaginemos que tenemos un estilo, llamado "MiEstilo" en el que el nombre de los autores se escribe en letra normal, y queremos que se escriba en negrita. Debemos abrir el fichero «`MiEstilo.dbj`» y buscar la pregunta correspondiente al formato del nombre del autor. Una vez localizada veremos lo siguiente:

```
%TYPEFACE FOR AUTHORS IN LIST OF REFERENCES:
    %: (def) Normal font for author names
% nmft,nmft-sc,%: Small caps authors
% nmft,nmft-it,%: Italic authors
% nmft,nmft-bf,%: Bold authors
% nmft,nmft-def,%: User defined author font
```

Para conseguir que los autores se escriban en negrita, simplemente debemos quitar la primera marca de comentario en la línea 5ª, y, tras ello, volver a ejecutar

```
latex MiEstilo.dbj
```

Lo que nos generará un nuevo fichero de estilo llamado «`MiEstilo.bst`» idéntico al anterior salvo en el hecho de que ahora los nombres de los autores se imprimen en negrita.

En ocasiones la activación de ciertas características puede implicar que haya a su vez que modificar la respuesta dada a alguna otra. Por ejemplo: si se eligió un estilo de cita numérico, y luego se quiere cambiar al estilo Autor-año, no basta con cambiar esta respuesta en el fichero «`.dbj`», sino que el estilo Autor-año implica que se habrían planteado otras preguntas cuya respuesta habrá que indicar en el fichero «`.dbj`».

5. Paquetes LaTeX relacionados con BibTeX

En la CTAN existen numerosos paquetes que añaden utilidades a BibTeX o que de alguna manera amplían las posibilidades del sistema.

No me es posible ocuparme de todos, y por ello seleccionaré los que me parecen más interesantes.

5.1. Referencias bibliográficas completas en el lugar donde son citadas

En determinadas especialidades existe la costumbre de incluir los datos completos de una referencia bibliográfica la primera vez que es citada en el texto. Hay dos paquetes dirigidos a obtener ese efecto: bibentry y footbib. Véamoslos por separado.

El paquete bibentry: Este paquete, que en la CTAN se incluye en el mismo directorio que «`natbib`», ha sido desarrollado por Patrick W. Daly. Con él se pretende que cuando se incluya en el texto una cita, se imprima no sólo la etiqueta representativa de la misma, sino la referencia bibliográfica completa.

El paquete incorpora los siguientes comandos LaTeX:

- "`\nobibliography{BaseDatos}`" Indica el nombre de la base de datos de la que debe extraer sus referencias el comando "`\bibentry`". Por lo tanto equivale al comando "`\bibliography`" con la salvedad de que no imprime una lista de referencias. De hecho "`\nobibliography`" debe estar definido antes de que se puedan usar los comandos "`\bibentry`"
- "`\bibentry{clave}`" Imprime la referencia bibliográfica completa en el lugar donde se encuentre, salvo el punto con el que la referencia termina en los estilos estándar. Antes del uso de "`\bibentry`", debe haberse indicado la base de datos de la que se extraerá el registro, mediante el comando "`\nobibliography`". Asimismo, una referencia citada por este procedimiento no será incluida en la lista de referencia final (que, si se quiere generar, debe ser por el procedimiento normal).

Aunque no se si es por alguna circunstancia debida a mi instalación, o por un defecto en el diseño del paquete, lo cierto es que el uso de este paquete me genera errores (no críticos, pero errores), y, además, afecta al formateo de la página donde se debe imprimir la referencia completa.

El paquete footbib: Este paquete ha sido diseñado por Eric Domenjoub. Define un comando llamado "`\footcite`", similar a "`\cite`", pero que provoca que la referencia en cuestión se imprima al pié de la página donde se hizo, sin que estas

notas entren en conflicto con las notas a pie de página normales. Asimismo los comandos "`\cite`" y "`\nocite`" pueden seguir usándose al modo habitual.

Las referencias generadas por "`\footcite`" usan su propia base de datos y estilo bibliográfico, cuyos nombres le son indicados mediante "`\footbibliography`" y "`\footbibliographystyle`". El paquete incluye otros comandos entre los que se puede citar "`\footcite*`" (que pone la marca en el texto pero no imprime la nota) y "`\footnocite`" (que imprimer la nota, pero sin poner marca alguna en el texto).

Lo que menos me gusta de este paquete es que las notas generadas por él mantienen una numeración distinta del resto de las notas a pie y aunque las distintas opciones del paquete ofrecen varias posibilidades respecto a su formateo, lo cierto es que cuando las notas generadas por este paquete coinciden con las notas normales, el resultado estético no es bueno.

5.2. Generación de varias listas bibliográficas

Existen varios paquetes para generar múltiples listas bibliográficas: bibtopic[46], bibunits, chapterbib[47], compactbib[48], multibbl[49] y multibib. De todos ellos, a mi modo de ver, los más interesantes son bibunit y multibib. El primero permite generar varias listas bibliográficas atendiendo al lugar del documento donde se produce la cita, y el segundo permite hacerlo atendiendo al contenido de la cita.

El paquete bibunit: Este paquete, elaborado por José Alberto Fernández, y modificado por Thorsten Hansen, permite trabajar con varias listas bibliográficas para distintas unidades de un documento. Una vez cargado el paquete, podemos usar el entorno bibunit cuyo formato es el siguiente:

```
\begin{bibunit}[Estilo]

  ... (texto normal)

  \putbib[NombreBase]
\end{bibunit}
```

[46] Permite generar listas bibliográficas diferentes, a partir de bases de datos distintas. Se generará una lista para cada uno de los ficheros «`.bib`» implicados.

[47] Genera listas bibligráficas independientes para cada fichero que haya sido incluido en el documento principal mediante el comando "`\include`".

[48] Este paquete no está diseñado para trabajar con BibTeX, sino con el entorno estándar de LaTeX «`thebibliography`».

[49] Fue diseñado pensando en referencias bibliográficas en idiomas que, al manejar distintos alfabetos, no tenga sentido la ordenación alfabética. Pero en la práctica sirve para generar cuantas listas bibliográficas deseemos.

Donde *Estilo* es el nombre de alguno de los estilos estándar de BIBTEX, y *NombreBase* es el nombre de nuestra base de datos. El resultado será que en el lugar donde se encuentre el comando "`\putbib`" se generará una lista de referencias con las obras correspondientes a todos los comandos "`\cite`" y "`\nocite`" incluidos dentro del entorno «`bibunit`», que no impide que también pueda generarse una lista de referencias global para el documento, mediante el procedimiento normal en BIBTEX.

El paquete también implementa el comando "`\bibliographyunit[Unidad]`" Donde *Unidad* (que debe ser introducido entre corchetes, no entre llaves), puede asumir los valores "`\chapter`" o "`\section`". El efecto de este comando es el siguiente:

1. Para cada unidad de las indicadas en el comando (capítulos o secciones) se generará una lista bibliográfica, si dentro de dicha unidad se incluye un comando "`\putbib`". La lista se incluirá precisamente donde se encuentre tal comando.

2. Los comandos "`\bibliography*`" y "`\bibliographystyle*`" indican la base de datos y el estilo para estas listas de referencias. Pero a falta de estos comandos, se usarán los comandos normales para trabajar con BIBTEX: "`\bibliography`" y "`\bibliographystyle`".

3. "`\bibliographyunit`" es compatible con los entornos «`bibunit`». Y si se encuentran los comandos "`\bibliography*`" y "`\bibliographystyle*`", la base de datos y el estilo allí indicados se usarán también para los entornos bibunits.

Por último, este paquete incluye también el comando "`\cite*`" cuyo efecto es incluir una referencia tanto en la lista de referencias global como en la local en donde se encuentre.

El paquete multibib: Este paquete añade lo que el anterior no tenía. Mediante el anterior paquete podíamos generar distintas listas bibliográficas atendiendo al *lugar* donde las citas tengan lugar, pero no había manera de diferenciar las obras que se incluirán en cada lista atendiendo a la materia de que cada obra trate. Multibib sí nos permite hacer eso.

Para entender bien el funcionamiento de este paquete empezaré por recordar lo que ya sabemos: Los cuatro comandos de LATEX fundamentales para la generación de una lista bibliográfica son "`\cite`", "`\nocite`", "`\bibliography`" y "`\bibliographystyle`". Pues bien: este paquete define un comando llamado "`\newcites`" cuyo formato es el siguiente:

```
\newcites{Sufijo}{Título}
```

y cuyo efecto es el de generar cuatro comandos adicionales cuyos nombres se componen del nombre de uno de los cuatro comandos que acabo de mencionar,

seguido del sufijo recibido como argumento. Es decir: se generarán cuatro comandos cuyos nombre serán: "`\cite[Sufijo]`", "`\nocite[Sufijo]`", cmd bibliography[Sufijo] y "`\bibliographystyle[Sufijo]`". La conjunción de los cuatro comandos generará una lista de referencias cuyo título será el del segundo argumento recibido por "`\newcites`".

Por ejemplo, en el presente documento se citan trabajos sobre BibTeX y otros que versan sobre otras cuestiones pero que, por las razones que sean he decidido citar. Si quisiera que las referencias bibliográficas se ordenaran atendiendo a ese criterio debería escribir, preferentemente al principio del documento:

```
\newcites{bib}{Bibliografía específica sobre BibTeX}
\newcite{gen}{Otras obras citadas}
```

Ello generará los siguientes ocho nuevos comandos:

1. "`\citebib`".
2. "`\nocitebib`".
3. "`\bibliographybib`".
4. "`\bibliographystylebib`".
5. "`\citegen`".
6. "`\nocitegen`".
7. "`\bibliographygen`".
8. "`\bibliographystylegen`".

Cada uno de estos comandos funciona de modo similar al comando *normal* equivalente, pero con la salvedad de que cada grupo de comandos sólo se relaciona entre ellos, es decir: "`\bibliographygen`" insertará una lista de referencias bibliográficas, cuyo título será "Otras obras citadas", construida con el estilo indicado en "`\bibliographystylegen`" y compuesta de las referencias indicadas mediante los comandos "`\citegen`" y "`\nocitegen`".

Para que esto funcione téngase en cuenta que para cada lista de bibliografía se generará un fichero «`.aux`» distinto, y cada uno de esos ficheros debe ser procesado con BibTeX.

5.3. Otros paquetes

Además de los paquetes mencionados, para trabajar con bibliografía están disponibles los siguientes paquetes[50]:

authorindex: Un paquete para generar un índice de autores citados en el documento con una lista de las páginas donde cada autor es citado.

[50] No se incluye en la siguiente lista, ni los paquetes que se limitan a instalar estilos bibliográficos adicionales, ni los relativos a la compatibilidad con babel, que ya han sido tratados.

bibarts: Generación de listas bibliográficas propias del campo de las artes.

bibcheck: En entornos «`thebibliography`» escritos a mano, comprueba que todos los elementos del mismo efectivamente hayan sido citados en el documento.

breakcites: Ligeras modificaciones al comando "`\cite`" para permitir saltos de línea dentro de una cita múltiple.

chbibref: Introduce un comando para cambiar el título de la lista de referencias. Este comando funciona en cualquier tipo de documento.

cite: Este paquete permite que cuando una cita incluye tres o más referencias consecutivas (en la lista de referencias), la cita aparezca *comprimida*. Por ejemplo: [1–4,6]. Implementa también un comando cmd citen- que hace que las citas aparezcan en el texto sin corchetes. Soporte para listas de citas numéricas comprimidas y ordenadas.

compactbib: En documentos con varias listas bibliográficas, permite una numeración única en todas ellas.

coverpage: Genera una página de cubierta para los documentos científicos, en la que se incluyen datos de BIBTEX.

din1505: Un estilo bibliográfico para textos alemanes, que trabaja con «`natbib`».

doipubmed: Introduce nuevos comandos para la gestión de la bibliografía.

gloss: Creación de glosarios usando BIBTEX.

jurabib: Bases de datos de BIBTEX para textos legales alemanes.

notoccite: Previene la numeración errónea de las citas cuando se usa el estilo «`unsrt`».

rangecite: Permite citar, en el documento, un rango de obras de la lista bibliográfica numerada.

tocbibind: Incluye los índices y la lista de referencias en el índice sistemático.

6. Otras cuestiones de interés

6.1. El comando newblock y la opción openbib

Para BIBTEX en una referencia bibliográfica los distintos campos se agrupan en *bloques*. Y así hay un bloque para la autoría, otro para el título, otro para los datos adicionales, etc.

Los estilos estandar, al formatear la lista de referencias bibliográficas, insertan el comando "\newblock" cada vez que acaba un bloque y empieza otro. Véase, por ejemplo, la siguiente referencia, tal y como el estilo «plain» la formatearía:

```
\bibitem{kopla}
Helmut Kopka and Patrick~W. Daly.
\newblock {\em Guide to {L}a{T}e{X}}.
\newblock Addison-Wesley, 4 edition, 2004.
```

Por defecto el comando "\newblock" no hace nada. Pero podemos redefinirlo para que haga algo; siendo este uno de los pocos casos en los que el funcionamiento de BIBTEX es, en cierto modo, controlable desde el documento «.tex».

Existe incluso una opción para los documentos «book» y «article» lamada «openbib» (que debe ser establecida en el comando "\documentclass") que provoca una redefinición de "\newblock". Cuando esta opción está activa cada bloque dentro de una referencia empezará en una línea nueva, la cual tendrá un sangrado igual al valor de "\bibindent", que por defecto es de 1.5 em.

6.2. Elementos @preamble en los ficheros «.bib»

En un fichero «.bib» podemos introducir un elemento «@Preamble» de acuerdo con el siguiente formato:

```
@Preamble{"  Cadena  de  texto  "}
```

y su efecto es el de escribir, al principio del fichero «.bbl» generado por BIBTEX, el texto entrecomillado. Esto es útil para introducir determinados comandos LATEX que provoquen algún tipo de funcionamiento especial. Como ejemplos pueden verse el mencionado en [3] para alterar el sistema de construcción de etiquetas en el estilo bibliográfico "alpha", o el mencionado en [10] para conseguir que en la lista bibliográfica dos registros se ordenen de cierta manera, alterando las reglas generales sobre ordenación de registros.

Un ejemplo más sencillo: Si queremos que en todos los documentos en los que se use nuestra base de datos el título de la lista de referencias bibliográficas sea "Bibliografía citada" y que automáticamente la bibliografía se incorpore al índice de contenido, podríamos escribir en el fichero «.bib», el siguiente texto:

```
@Preamble{ "\renewcommand{\refname}{Bibliografía citada} "
# "\addcontentsline{toc}{section}{Bibliografía citada} " }
```

También es interesante el ejemplo que recoge [8]:

```
@preamble{ "\makeatletter" }
@preamble{ "\@ifundefined{url}{\def\url#1{\texttt{#1}}}{}" }
@preamble{ "\makeatother" }
```

Este ejemplo serviría para asegurarnos de que en el contenido de los campos de nuestra base de datos podemos usar el comando "`\url`", aunque la base se vaya a usar en un fichero «`.tex`» en el que no se haya cargado el paquete «`url`» ni se haya definido de ninguna manera dicho comando.

En este último ejemplo se han usado tres elementos preamble, en lugar de un solo elemento que conste de varias cadenas concatenadas (como se hizo en el ejemplo anterior). En teoría hacer una cosa u otra da igual, pero hay que tener en cuenta que el autor de BIBTEX afirma en [10] que existe un límite respecto del número de comandos "`Preamble`" que se pueden usar, pero no aclara cuál es ese límite. En todo caso, puesto que en preamble puede usarse la concatenación de cadenas (como se muestra en el primero de los ejemplos), en realidad con usar una sola función "`Preamble`" será siempre suficiente[51].

6.3. Comentarios y registros desconocidos en los ficheros «`.bib`»

De lo hasta ahora visto se desprende que el formato interno de un fichero «`.bib`» es siempre:

```
@NombreElemento{ContenidoElemento}
```

Donde NombreElemento puede ser "String", "Preamble" o el nombre de los distintos tipos de registro previstos para BIBTEX, y ContenidoElemento depende del elemento de que se trate.

Esto es cierto, pero a ello hay que añadir que el fichero será válido aunque además de ese contenido tenga cualquier otro. Y ese otro contenido no debe, además ajustarse a ningún formato específico: simplemente será ignorado por BIBTEX, pero no impedirá que el contenido válido sea localizado y usado.

Podemos pues incluir un texto explicativo de cada tipo de registros, o, si lo preferimos, el capítulo primero del Quijote. ¡¡¡Podemos escribir lo que queramos!!! Sin sujetarnos a ninguna regla. Así, por ejemplo, el siguiente fichero:

```
# Fichero de prueba para BibTeX
# Las líneas de este texto van precedidos del signo "#"
# aunque no les hace falta. Ese signo, además, es
# interpretado por  BibTeX como concatenador de cadenas
# de texto. Pero no genera ningún error.

@Book{knuth,
   author = {Donald E. Knuth},
   title = {The {TeX} {Book}},
   year = 1986,
```

[51] De hecho los estilos de BIBTEX ejecutan la función interna «`preamble$`» que hace que todos los elementos «`@preamble`» del fichero «`.bib`» se concatenen. Por ello, cuando usemos estos elementos, conviene tener en cuenta la posible concatenación, para dejar los espacios en blanco suficientes al principio y al final de la cadena en que consiste el elemento.

```
   publisher = {Addison-Wesley},
}

Este libro contiene, en palabras de su autor, "todo lo que hay que
saber sobre TeX". Desgraciadamente no está disponible en España,
aunque podemos comprarlo a través de Amazon.com.

@libro{jqtaz,
   autor = {Joaquín Ataz López},
   titulo = {Libro de prueba},
   fecha = 2006,
}
```

Se verá que se ha incluido un registro válido, precedido de un texto y seguido por otro texto. También hay un registro de tipo inexistente para BIBTEX. Ni el texto precedente ni el posterior al registro válido impiden que BIBTEX localice el registro correcto. Y tampoco el registro inexistente genera ningún error[52].

Por esa razón en BIBTEX no es preciso definir ningún carácter para comentarios: Estos se pueden introducir libremente. No obstante, en su especificación oficial viene prevista la existencia de un elemento «`@Comment`» que existe exclusivamente para mantener la compatibilidad entre los ficheros BIBTEX y los ficheros Scribe.

[52] De hecho, al menos en los estilos estándar, no existen propiamente hablando registros inexistentes, pues todo registro cuyo nombre no sea identificado como un tipo de registro válido, será formateado como si fuera un registro de tipo «`Misc`». Y en el ejemplo que he puesto, si en el registro de tipo «`@Libro`» los campos en lugar de «`autor`», «`titulo`» y «`fecha`» fueran «`author`», «`title`» y «`year`», el registro sería correctamente formateado, no como un libro, pero sí como un registro «`Misc`».

Parte III
BIBTEX avanzado

7. Los ficheros de estilo como programas

Los ficheros de estilo manejados por BIBTEX son en realidad *programas* escritos en un lenguaje innominado que informan a BIBTEX acerca de cómo deben formatearse los datos leídos de la base de datos. Como el lenguaje no tiene nombre oficial, cuando me tenga que referir a él lo denominaré BST que es la extensión utilizada por los ficheros de estilo[53].

7.1. Cuestiones generales

7.1.1. Reglas generales de sintaxis

Las características generales de BST son las siguientes:

1. No se distingue entre mayúsculas y minúsculas *para los elementos del lenguaje*, entendiendo por tales los nombres de los comandos, variables y funciones, así como las cadenas de texto usadas como patrón en determinadas funciones internas. Por el contrario, en el contenido de los campos y de las variables de texto, sí se distinguen las mayúsculas de las minúsculas.

2. Los saltos de línea y los tabuladores se tratan como espacios en blanco, y dos o más espacios en blanco se tratan como si fueran sólo uno. Y a diferencia de otros lenguajes no existe ningún carácter reservado para indicar el fin de una instrucción: ni el punto y coma (como en C o Pascal) ni el salto de línea (como en Basic y similares). Los saltos de línea se usan exclusivamente para hacer más comprensible para los seres humanos lo que una función hace.

   No obstante lo anterior, de cara a la información que BIBTEX emite cuando se produce algún error (y que se almacena en el fichero «`.blg`» generado tras la ejecución de BIBTEX), se recomienda:

   a) Insertar una o más líneas en blanco entre la definición de dos funciones.

   b) No introducir líneas en blanco dentro de una función.

   Lo anterior, por supuesto, no se aplica al contenido de las variables de texto. En ellas todos los espacios en blanco son relevantes y se distingue entre

[53] Eso no significa que el lenguaje se llame BST. Simplemente, si no uso un nombre para referirme a él, en las próximas páginas tendría que emplear demasiados circunloquios.

espacios en blanco y tabuladores[54]. Asimismo las constantes de texto, que se introducen entrecomilladas, deben escribirse en la misma línea: entre las comillas de apertura y cierre de una cadena de texto no puede haber un salto de línea.

3. Se puede hacer uso de las llaves para agrupar bloques de código.

4. El carácter "%" está reservado para los comentarios: Todo lo que se escriba a la derecha del mismo, hasta el final de la línea, será ignorado.

7.1.2. Estructura de los programas BST

Básicamente un programa BST sigue la siguiente estructura:

1. En primer lugar se encuentra la *definición* de los campos con los que se va a trabajar. Ello se hace mediante el comando ENTRY. Cualquier campo existente en la base de datos que no se haya incluido en esta *lista de campos* no será leído ni cargado en memoria.

2. A continuación se encuentra la definición de una función para cada uno de los tipos de registro que puede haber en la base de datos. La función debe tener el mismo nombre que el tipo de registro de que se trate, y en ella se debe especificar cómo formatear los datos de ese tipo de registro. Debe asimismo definirse una función llamada «`default.entry`» que formateará los datos de los registros cuyo nombre no coincida con el de ninguna de las funciones anteriores. En los ficheros de estilo estándar de BIBTEX esta función se limita a considerar que todo registro que no sea de algún tipo predefinido es un registro de tipo «`Misc`». Por otra parte, como las reglas del lenguaje obligan a que no se puedan invocar variables o funciones que no hayan sido previamente definidas, la escritura de las funciones anteriores implicará normalmente la escritura de funciones adicionales y la declaración de variables.

3. También es posible añadir una lista de MACROS o abreviaturas predefinidas para la base de datos.

4. A continuación se encuentra el comando READ, que es el verdadero centro neurálgico de un programa BST. Este comando lee la base de datos y construye la lista de datos a partir de los datos leídos, cargando el contenido de los campos, para cada registro de la lista.

5. Una vez leídos los datos, y generada la lista que los contiene, viene la fase de comprobación, formateo (ejecutando las funciones de formateo que se han definido para cada tipo de registro) y ordenación de la lista.

[54] Ello con independencia de que si finalmente el contenido de la variable termina formando parte de una lista de referencias insertada en un documento «`.tex`», los espacios en blanco y tabuladores existentes en ella reciban el tratamiento que LATEX da a tales caracteres.

6. Por último la lista de datos se vuelca en el fichero «`.bbl`» correspondiente.

En la secuencia que acabo de indicar hay partes que se *suelen* hacer y otras que se *deben* hacer. Así, por ejemplo, he dicho que suele empezarse por el comando ENTRY que define la lista de campos. Esto no es obligatorio, lo único verdaderamente obligatorio es que el comando ENTRY sea anterior al comando READ y que los campos estén definidos antes de ser utilizados, es decir: antes de que alguna de las funciones de formateo de los distintos tipos de registro los lleguen a mencionar.

7.2. Objetos del lenguaje BST

Aunque el lenguaje BST dispone de los elementos presentes en la mayoría de los lenguajes de programación (variables, funciones, etc). En un primer nivel de aproximación, el lenguaje sólo consta de comandos. Y los comandos existentes son solamente 10, los cuales se usan para generar o definir el resto de los elementos del lenguaje.

7.2.1. Los comandos de BST

Podemos agrupar a los comandos atendiendo a diferentes criterios. Por orden alfabético sus nombres son ENTRY, EXECUTE, FUNCTION, INTEGERS, ITERATE, MACRO, READ, REVERSE, SORT y STRINGS. El formato para los comandos es el siguiente:

```
NombreComando [{Argumento1}] [{Argumento2}] [{Argumento3}]
```

Donde *NombreComando* es alguno de los diez comandos que he especificado (en mayúsculas o minúsculas, es indiferente) y tras el nombre, entre llaves, se debe introducir los argumentos que el comando espera. No todos los comandos reciben argumentos, ni todos reciben el mismo número de argumentos. Hay dos que no reciben ningún argumento (READ y SORT), cinco que reciben un sólo argumento (EXECUTE, INTEGERS, ITERATE, REVERSE y STRINGS), dos que reciben dos argumentos (FUNCTION y MACRO) y uno que recibe tres argumentos (ENTRY).

Se aplican además las siguientes restricciones sobre el número y orden de los comandos:

1. Sólo puede haber un comando ENTRY y un comando READ. Los demás comandos pueden aparecer tantas veces como se desee.

2. El comando ENTRY y los comandos MACRO deben encontrarse antes del comando READ. También debe encontrarse antes de READ los comandos FUNCTION que declaren la acción básica ante cada uno de los posibles tipos de registro (véase el epígrafe anterior).

3. Los comandos EXECUTE, ITERATE, REVERSE y SORT deben necesariamente encontrarse después del comando READ.

En cuanto al significado y uso de los comandos, más adelante se explicarán la mayoría de ellos. Ahora me centraŕe en dos: READ y SORT:

READ: Es el comando central de un programa BST. Su efecto es el de leer la base de datos y generar la lista de datos, que contendrá, para cada uno de sus elementos, un valor para cada uno de los campos y de las variables de lista definidas mediante el comando ENTRY. Tanto el nombre de la base de datos a leer como los datos concretos que hay que leer se extraen del fichero «`.aux`» generado por LaTeX durante su compilación.

SORT: Tras la ejecución del comando READ, la lista de datos se encuentra en el orden en el que en el fichero «`.aux`» se encontraban las distintas citas. El comando SORT reordena la lista atendiendo al valor de la variable global «`sort.key$`».

7.2.2. La lista de datos

La tarea fundamental de BibTeX es leer una serie de datos, formatearlos y volcarlos, una vez formateados, en un determinado fichero. Por lo tanto la lista de datos es una noción esencial. Esta lista se genera cuando se ejecuta el comando READ. De la apertura del fichero que contiene la base de datos, localización y lectura de los registros precisos se ocupa automáticamente BibTeX, es decir: no hay que especificar ninguna rutina; el comando READ sabe lo que tiene que hacer.

Para cada registro de la base de datos que se haya leído habrá un *elemento* en la lista. Este elemento estará compuesto a su vez de los campos del registro y de las variables de lista que se hayan declarado (véase la sección 7.2.3). Estas variables de lista, por otra parte, inicialmente carecerán de valor: tendrá que ser alguna función la que se lo de.

En relación con los campos del registro leído pueden darse dos circunstancias:

1. Que en un registro concreto exista algún campo que no esté incluido en la lista de campos que el programa ha debido declarar antes de la lectura. En tal caso ese *campo* no previsto será ignorado y no se cargará en la lista de datos.

2. Que en el registro leído no existiera alguno de los campos declarados en el programa. Esto es bastante habitual porque el programa declara todos los campos posibles, pero no en todos los registros habrá un valor para cada campo. Cuando para un campo no hay datos en un registro, dicho campo adopta el valor de “omitido”. Un campo “omitido” no es lo mismo que un

campo vacío, ya que este último se produce cuando un campo *estaba* en el registro, pero su valor es una cadena vacía[55].

En la lista de datos no se incorporan en realidad los datos de la base de datos tal y como hayan sido leídos, sino que BIBTEX, al leer el registro, aplica automáticamente las abreviaturas definidas en la base de datos mediante elementos de tipo «`@String`», así como las macros definidas en el propio programa mediante el comando MACRO.

Este comando tiene el siguiente formato:

```
MACRO {NombreMacro} {"Texto de la Macro"}
```

y su efecto es el de crear una abreviatura que no está localizada en el fichero «`.bib`», sino en el propio estilo y, por lo tanto, se podrá usar en cualquier base de datos que utilice dicho estilo.

Las abreviaturas creadas con este comando son totalmente equivalentes a las generadas dentro del fichero «`.bib`» con «`@String`», hasta el punto de que si una abreviatura definida con MACRO tuviera el mismo nombre que una abreviatura definida con «`@String`», la segunda sobreescribiría a la primera.

Todos los comandos MACRO de un programa BST deben encontrarse antes del comando READ.

Si nuestro estilo maneja bases de datos en los que los meses se hayan escrito con las abreviaturas estandar en inglés, pero queremos que en la lista de referencias los nombres figuren en español, deberíamos[56] escribir en nuestro estilo:

```
MACRO {jan} {"Enero"}
MACRO {feb} {"Febrero"}
MACRO {mar} {"Marzo"}
MACRO {apr} {"Abril}
MACRO {may} {"Mayo}
...
```

7.2.3. Variables y constantes

Tipos de datos: Los datos sobre los que se trabaja, bien se introduzcan directamente en alguna sentencia del programa, bien se encuentran almacenados en alguna variable, pertenecerán necesariamente a uno de los siguientes dos tipos:

---

[55] La diferencia es exclusivamente a efectos de programación. Desde el punto de vista práctico es claro que una vez formateada la referencia, no hay diferencias entre un campo omitido y un campo vacío: el dato que dicho campo debía contener, no está en ninguno de ambos casos.

[56] En [3] se dice que los estilos *deben* incorporar macros para los nombres de los meses. Pero no es cierto que *deban hacerlo*, aunque posiblemente sea una buena idea hacerlo ya que si nuestro estilo se va a manejar con múltiples bases de datos, no debemos olvidar que en las recomendaciones generales sobre BIBTEX se suele incluir la de usar para los meses del año las abreviaturas estándar en inglés.

1. Datos numéricos: En este rango entran exclusivamente los números enteros. BST no puede trabajar directamente con números que tengan decimales.

2. Datos alfanuméricos: Consisten en una secuencia de cero o más caracteres. Una secuencia de cero caracteres es una cadena vacía. Dentro de una cadena de texto está admitido absolutamente cualquier carácter imprimible y los espacios en blanco son relevantes.

En la tipología de los lenguajes de programación se distingue entre lenguajes tipados y no tipados, dependiendo de si los tipos de los datos deben ser establecidos de antemano o si, por el contrario, el propio lenguaje dinámicamente es capaz de, analizando un dato, decidir a qué tipo pertenece. Pues bien BST es un lenguaje fuertemente tipado (aunque con pocos tipos de datos), lo que significa que en el momento de definir una variable ya hay que indicar si contendrá datos numéricos o alfanuméricos, al introducir valores constantes hay que indicar el tipo de los mismos y, además, no hay apenas funciones de conversión de tipos.

Variables: En BST se admiten dos tipos distintos de variables. Las variables globales y las variables de lista, cada una de las cuales, por su parte, puede ser numérica o alfanumérica, según el tipo de datos que pueda almacenar.

Las variables globales son, las que podríamos llamar *normales*. Tienen un solo valor que está accesible para cualquier función siempre y cuando la función se haya definido después de la declaración de la variable.

Junto con las variables globales se encuentran las variables de lista. En ellas la misma variable asume un valor diferente para cada uno de los elementos de la lista. Un caso concreto de variables de lista son los campos, que se consideran variables de lista especiales porque deben ser declarados expresamente como campos, para que, en el momento de la lectura de la base de datos, BIBTEX asigne a cada campo su valor correspondiente.

Pero junto con los campos podemos declarar las variables de lista que queramos. Al declarar una variable de lista es como si estuviéramos añadiendo un campo que sabemos que no está en la base de datos, pero que pensamos llenar mediante alguna función; normalmente la que formatee los datos del registro.

Por ejemplo: los cuatro estilos estándar de BIBTEX definen una variable de lista alfanumérica llamada «`label`» que se usa para almacenar la etiqueta identificativa de dicho registro, que será luego devuelta por el comando de LATEX "`\cite`".

Al igual que el resto de los datos manejados por BST las variables de lista deben ser numéricas o alfanuméricas. Los campos se consideran siempre alfanuméricos, y el tipo del resto de las variables de lista se define en el mismo momento en el que se declara la propia variable.

Declaración de variables: Para declarar variables se usan tres comandos:

STRINGS: Permite declarar variables globales alfanuméricas. Tras el comando hay que escribir, entre llaves, los nombres de las variables que se quieren declarar, separados por espacios en blanco o por saltos de línea. Por ejemplo, para declarar dos variables alfanuméricas llamadas "nombre" y "lista" habría que escribir la siguiente línea:

```
STRINGS { nombre lista }
```

INTEGERS: Permite declarar variables globales numéricas. Su formato es similar a STRINGS, y así, por ejemplo, la próxima línea declararía una variable numérica llamada "cont":

```
INTEGERS { cont }
```

ENTRY: Este comando es el que se usa para declarar las variables de lista, incluidos los campos. Su principal peculiaridad estriba en que en un fichero «`.bst`» sólo puede haber un comando ENTRY (que suele ser el primer comando del fichero). Su formato es:

```
ENTRY
   {Lista de campos}
   {Variables de lista numéricas}
   {Variables de lista alfanuméricas}
```

En "*lista de campos*" hay que incluir todos los campos de la base de datos con los que se piensa trabajar, con independencia de que se trate de campos presentes en todos los registros o sólo en alguno de ellos. Los nombres de los campos se deben separar entre sí por espacios en blanco (o saltos de línea).

En el caso de que no queramos usar variables de lista numéricas o alfanuméricas, hay que escribir las llaves sin nada entre ellas, pero no puede dejar de indicarse expresamente todos los argumentos esperados por el comando. Así en el siguiente ejemplo:

```
ENTRY { address author booktitle chapter edition editor
        howpublished institution journal key month note
        number organization pages publisher school series
        title type volume year }
   { }
   { label }
```

estariamos definiendo los campos de una base de datos estándar de BIBTEX, así como la variable de lista alfanumérica «`label`» y ninguna variable de lista numérica.

Los nombres de las variables deben empezar por una letra, y pueden contener cualquier carácter salvo espacios en blanco (o tabuladores) y los diez caracteres

reservados de LaTeX. Es muy normal que los nombres de las variables incluyan puntos, de tal modo que se consigan nombres verdaderamente descriptivos como en «`esto.es.una.variable.de.ejemplo`».

Variables predefinidas: Hay tres variables globales y un campo predefinidos (que no hay que declarar explícitamente):

1. «`sort.key$`»: Es una variable global alfanumérica usada para indicar al comando SORT el criterio de ordenación de los elementos que componen la lista de datos.
2. «`entry.max$`»: Es una variable global numérica que indica la longitud máxima de las variables de lista alfanuméricas. Hay que usarla si queremos limitar dicha longitud; el efecto será que la variable se truncará.
3. «`global.max$`»: Es una variable global numérica que indica la longitud máxima de las variables globales alfanuméricas. Si se le da algún valor, las variables globales alfanuméricas cuya longitud supere dicho valor, quedarán truncadas.
4. «`crossref`»: Es un campo implícito en la base de datos, es decir: no hay que declararlo expresamente. Se usa para las referencias cruzadas.

Constantes: En cualquier lugar donde se espere un dato podemos usar el nombre de una variable, en cuyo caso el dato a manejar sería el contenido de la misma, o un valor constante. Estos se representan de la siguiente manera:

1. Los valores constantes alfanuméricos se deben introducir entre comillas.
2. Los valores constantes numéricos se deben introducir precedidos del signo "#" y sin ningún espacio en blanco entre dicho signo y el número de que se trate. Y así «`#1`» es una constante válida, pero «`# 1`» no lo es.

7.2.4. Funciones

Hay dos tipos de funciones: las internas y las definidas por el usuario. Las funciones internas se verán más adelante, las definidas por el usuario se generan mediante el comando FUNCTION cuyo formato es

```
FUNCTION {NombreFunción}
  { Contenido de la función }
```

Donde para el nombre de la función se aplican las mismas reglas que para el nombre de las variables, es decir: debe empezar por una letra, y puede contener

cualquier carácter imprimible (lo que excluye espacios en blanco, tabuladores y saltos de línea) salvo cualquiera de los caracteres reservados de LaTeX.

En BST las funciones deben ser definidas antes de ser invocadas, y no se admite la recursividad (funciones que se invocan a sí mismas). Para invocar una función se pueden usar los siguientes comandos:

1. Mediante el comando «`EXECUTE {NombreFuncion}`». La función se ejecutará una sola vez.
2. Mediante los comandos «`ITERATE`» o «`REVERSE`» `{NombreFuncion}`.

Los dos comandos mencionados en último lugar ejecutan la función recibida como argumento en cada uno de los elementos de la lista de datos. La diferencia entre ambos estriba en el orden en el que estos elementos son tratados: ITERATE ejecuta la función empezando por el primer elemento de la lista y terminando por el último, mientras que REVERSE lo hace en el orden inverso.

Cuando desde una función se quiere invocar a otra, no es preciso (ni posible) usar ninguno de estos comandos, sino que basta con escribir su nombre, y así en el siguiente ejemplo:

```
FUNCTION {ejemplo.llamada.funciones}
  {
     llamar.funcion.1 llamar.funcion.2
     llamar.funcion.3
}
```

La función que he denominado «`ejemplo.llamada.funciones`» se limita a llamar (y ejecutar) tres funciones, llamadas, respectivamente, «`llamar.funcion.1`», «`llamar.funcion.2`» y «`llamar.funcion.3`»[57].

7.2.5. La pila del programa

Al explicar las variables no he dicho cómo se les asigna valor. Y al explicar las funciones se ha visto que ni reciben argumentos ni devuelven valores. Ello es porque tanto la asignación de valores a las variables, como el intercambio de datos entre funciones se realiza a través de la pila del programa[58], que es una zona de

[57] Obsérvese como el salto de línea que separa a la segunda función de la tercera, podría ser un espacio en blanco. En BST las líneas no se corresponden con las instrucciones, y se usan sólo para aumentar la legibilidad del programa.

[58] En teoría podríamos usar las variables globales para comunicar entre sí a las funciones. Pero esto no es casi nunca una buena idea, como, por otra parte, dicen casi todos los manuales de técnicas de programación. El uso indiscriminado de variables globales suele ser fuente de errores muy difíciles de localizar. Además, como para el uso de las funciones internas no hay más remedio que usar la pila, no tiene demasiado sentido rehuir dicho uso en nuestras propias funciones.

memoria en la que se puede ir *apilando* datos (de ahí el nombre de *pila*). Podemos imaginar que la pila es un cajón en el que se van almacenando carpetas (los datos). Las carpetas se van dejando una sobre otra en el cajón, de tal manera que en cada momento la carpeta superior es siempre la última que se introdujo. Cada carpeta que añadimos aumenta el tamaño del montón, y cada carpeta que extraemos lo reduce.

Las operaciones básicas en una pila son siempre introducir datos y extraer datos y en ellas el orden es inalterable. No se puede acceder a un dato que esté en la pila sin sacarlo de ella, y no es posible sacar un dato sin antes haber sacado todos los que estaban sobre él, por haber sido introducidos después; es decir: para sacar de nuestro cajón una determinada carpeta es preciso sacar antes todas las carpetas que estaban sobre ella[59].

En BST las funciones usan la pila de la siguiente manera:

- Antes de llamar a la función se dejan en la pila los datos que esta necesitará (lo que en otros lenguajes de programación equivaldría a sus parámetros).
- La función se ocupa de extraer esos datos de la pila, de tal manera que tras la ejecución de ésta podemos asumir que en la pila ya no estarán los datos que se introdujeron en ella.
- Si la función tiene que devolver algún resultado, lo hará introduciéndolo en la pila.

Dependiendo de lo que la función haga, en la pila podemos querer introducir:

1. El valor almacenado en una variable. Para ello basta con escribir el nombre de la variable. El nombre de una variable escrito dentro de una función BST tiene como efecto el enviar a la pila el valor de la misma.
2. Un valor constante, por ejemplo, el número “5”, o el texto “tomo”. Para enviar valores constantes se aplican las reglas que se explicaron en la página 59 sobre constantes, es decir: los valores numéricos se escriben precedidos del signo “#” y los alfanuméricos se entrecomillan.
3. El nombre de una variable, no su valor. Esto se hace sobre todo con las funciones que asignarán algún valor a la variable. Para introducir en la pila el nombre de la variable y no su valor, hay que preceder el nombre de un apóstrofe, sin dejar ningún espacio en blanco entre el apóstrofe y el nombre. Y así mientras, por ejemplo, «`cont`» envía a la pila el valor de la variable así

[59] Lo que en terminología informática se indica diciendo que las pilas son estructuras de datos de tipo LIFO, siendo esta última palabra las siglas en inglés de la frase: “*Last In, First Out*”, que en español significa que “el último en entrar es el primero en salir”. O sea: el orden de extracción de datos es exactamente el inverso al orden en el que los datos se introdujeron.

llamada, «`'cont`» lo que envía a la pila es el nombre de la variable, no su valor.

4. El nombre de una función. Esto se hace con algunas funciones que pueden ejecutar otras funciones, cuyo nombre deben conocer. Para introducir en la pila el nombre de una función se hace igual que para introducir el nombre de una variable: se escribe un apóstrofe precediendo al nombre. Eso hace que BST en lugar de ejecutar la función, envíe a la pila su nombre.

De lo que se acaba de decir podemos además inferir que a diferencia de otros lenguajes de programación, en BST no hay en sentido estricto sentencias, sino instrucciones que se ejecutan inmediatamente; y si normalmente las instrucciones se escriben en líneas, ello es para dotar de mayor claridad al programa. podríamos escribir enteramente una función sin que en ella hubiera ningún salto de línea, y podríamos también usar una línea para cada palabra.

Aunque lo mejor para ilustrar lo que se acaba de decir es un ejemplo. Imaginemos que tenemos una variable numérica llamada «`contador`» y que queremos incrementarla en dos elementos. En los lenguajes de programación *normales* ello se podría escribir de muchas maneras como:

```
contador += 2;
contador = contador + 2;
contador := contador + 2;
```

En BST habría que escribir:

```
contador #2 + 'contador :=
```

Analicemos esta línea para entender bien cómo funciona la pila:

1. La primera palabra “contador” es el nombre de la variable con la que queremos trabajar, y el efecto de escribirla es el de que su valor se almacenará inmediatamente en la pila (sin esperar a que acabe la linea).

2. A continuación hemos escrito “#2”, eso provoca que se envíe a la pila el valor numérico “2”. El carácter “#” que precede al “2” es necesario, de acuerdo con las reglas sobre la introducción de constantes que antes expliqué.

3. A continuación hemos escrito el carácter “+”, el cual es el nombre de una función interna de BST cuyo efecto es extraer de la pila los dos últimos elementos (que deben ser numéricos), sumarlos, e introducir en la pila el resultado de la suma. Tras haberse ejecutado esta función, en la pila ya no estará ni el valor de “contador” ni el número “2”, ya que ambos han sido *extraidos* de la pila. En la pila ahora estará, tan solo, el resultado de la función “+”: la suma de ambos valores.

4. Después hemos escrito *'contador*, es decir el nombre de la variable con la que estamos trabajando, precedido de un apóstrofe. El apóstrofe indica que no queremos almacenar en la pila el valor de la variable, sino exclusivamente su nombre.

5. Por último ":=" es otra función interna de BST cuyo efecto es extraer de la pila los dos últimos valores introducidos, asignar al primer elemento extraído (último que se introdujo) el valor del segundo elemento extraído (penúltimo que se introdujo), para ello se asume que el último elemento introducido debió ser el nombre de una variable (su dirección en memoria) y el penúltimo un valor adecuado para dicha variable, es decir: del mismo tipo (numérico o alfanumérico) que ella. Esta función no introduce nada en la pila.

¿Complicado? No demasiado, sólo es cuestión de acostumbrarse a que en BST las cosas que escribimos tienen un efecto inmediato: Si son variables o constantes se introducen en la pila, y si son funciones, actúan sobre los datos en la pila y eventualmente introducen nuevos datos en ella, que podemos leer para comprobar el resultado de la función.

No obstante, como estar continuamente refiriéndonos a la pila es muy tedioso, en adelante me referiré a las funciones como si la pila no existiera, en el bien entendido de que cuando digo que una función recibe como parámetro un dato, dicho dato lo recibirá a través de la pila. Asimismo cuando diga que una función devuelve cierto valor, quiero decir que dicho valor será introducido en la pila y para comprobarlo habrá que extraerlo de ella. Y en fin: cuando una función deba recibir varios datos, me referiré a ellos siempre en el orden en el que se introdujeron, y no en el orden en el que la función los debe extraer (que es el inverso).

7.3. Operadores y estructuras de control

BST carece de algunos elementos que otros lenguajes de programación tienen tales como operadores o estructuras de control. No obstante existen funciones internas de BST que cumplen dicho papel. A continuación me ocuparé de ellas:

7.3.1. Operadores

Operadores de comparación: Estos operadores son tres: '<', '>' y '='. Todos ellos reciben dos datos que deben ser comparados. En '=' los datos pueden ser numéricos o alfanuméricos, en '<' y en '>' deben ser necesariamente numéricos.

'<' Devuelve un "1" si el primer dato es menor que el segundo. En caso contrario devuelve un "0".

'>' Devuelve un "1" en la pila si el primer dato es mayor que el segundo, en el caso contrario devuelve un "0".

‘<’ Devuelve un “1” en la pila si ambos datos son iguales, en el caso contrario devuelve un “0”.

Operadores matemáticos y de concatenación: Los operadores matemáticos son ‘+’ y ‘−’, ambos actúa sobre dos datos numéricos. El operador de concatenación es ‘*’ y actúa sobre dos datos alfanuméricos:

‘+’ Recibe dos números y devuelve la suma de ambos.

‘−’ Recibe dos números y devuelve el resultado de restar el segundo del primero.

‘*’ Recibe dos cadenas de texto y devuelve una cadena igual a la concatenación de ambas. La concatenación se hace en el orden en el que las cadenas originales fueron insertadas en la pila.

El operador de asignación: Para asignar cualquier valor a cualquier variable se usa el operador de asignación, cuyo formato es el siguiente:

```
Valor 'NombreVariable :=
```

Donde «`Valor`» debe ser adecuado al tipo de variable de que se trate, es decir: un número si es variable numérica o una cadena de texto si es variable alfanumérica, y «`'NombreVariable`» es el nombre de una variable previamente definida por cualquiera de los procedimientos que permiten definir variables (comandos ENTRY, INTEGERS o STRINGS).

Este operador no introduce ningún dato en la pila.

7.3.2. Estructuras de control

En BST no hay, en el sentido estricto de la palabra, estructuras de control, es decir: instrucciones que permitan desviar el flujo del programa dependiendo de si se produce o no alguna condición. Pero lo que si hay es dos funciones internas que pueden usarse para ese objetivo.

La función if$: Recibe tres datos: Un entero y el nombre de dos funciones, o dos bloques de código. Si el entero es mayor que cero, ejecutará la primera función o bloque de código, y si es igual o menor que cero ejecutará la segunda.

El primer dato recibido suele ser fruto de una comparación realizada mediante alguno de los operadores de comparación que se acaban de explicar. Por lo tanto el esquema general del uso de «`if$`» es el siguiente:

```
   Comparación
   { Bloque a ejecutar si la comparación es cierta }
   { Bloque a ejecutar si la comparación no es cierta }
if$
```

Por ejemplo, el siguiente bloque de código, extraído de los estilos estándar:

```
output.state mid.sentence =
  { ", " * write$ }
  { output.state after.block =
      { add.period$ write$
        newline$
        "\newblock " write$
      }
      { output.state before.all =
          'write$
          { add.period$ " " * write$ }
        if$
      }
    if$
    mid.sentence 'output.state :=
  }
if$
```

En la primera línea se compara el valor de las variables «`output.state`» y «`mid.state`». Si ambas son iguales se ejecutará el bloque de código contenido en la segunda línea. Pero si son distintas se ejecutará el bloque contenido entre la tercera y la penúltima línea, el cual, a su vez, contiene nuevas funciones «`if$`» anidadas.

La función while$: Esta función recibe el nombre de dos funciones (o bloques de código) que va ejecutando alternativamente hasta que la segunda función devuelva un valor igual o menor que cero. En ella se usan las llaves para agrupar bloques de código de modo similar a como se hace en «`if$`».

While$ se usa fundamentalmente de dos maneras:

1. Para repetir cierto número de veces un bloque de código, de modo similar a lo que en otros lenguajes de programación hace la estructura de control "for".

2. Para repetir un bloque de código mientras se cumpla cierta condición.

7.4. Las funciones internas del lenguaje

Para la explicación del resto de las funciones internas, hará como he hecho para los operadores y estructuras de control: obviaré el uso de la pila salvo cuando sea imprescindible para la comprensión. Pero aunque no se diga expresamente hay que tener en cuenta que la pila se usa para enviar parámetros a las funciones y, en su caso, para que estas envíen el resultado de sus operaciones.

add.period$: Recibe una cadena de texto, si su último carácter, sin contar las llaves que la cadena pueda tener, es distinto de un punto o de un signo de cierre de interrogación o exclamación, le añade un punto al final. Devuelve la cadena modificada.

call.type$: Ejecuta la función cuyo nombre coincide con el tipo de registro correspondiente a un elemento de la lista de datos. Y así, por ejemplo, para un elemento de dicha lista que sea de tipo "book" se ejecutará la función "book" que debe haberse definido antes del comando READ. Si no existiera ninguna función prevista para ese tipo de registros, se ejecutaría la función denominada "default.type", que también debe haber sido definida antes del comando READ. Esta función normalmente se usará como argumento de un comando ITERATE o REVERSE.

change.case$: Recibe una cadena de texto y una cadena de formato. Y modifica la primera atendiendo al contenido de la segunda. Devuelve la cadena modificada. La cadena de formato puede ser:

- 'T' ó 't': Se pone en minúsculas la segunda cadena entera salvo su primera letra, y el primer carácter no blanco posterior a un punto.
- '1': Se pone en minúsculas toda la segunda cadena.
- 'U' ó 'u': Se pone en mayúsculas toda la segunda cadena.
- Cualquier otro contenido: la segunda cadena se deja intacta.

Los caracteres de la segunda cadena encerrados entre llaves, no se verán afectados por la transformación. Asimismo si cualquiera de los datos extraidos resultara ser numérico, se devuelve una cadena vacía.

chr.to.int$: Recibe un carácter individual y devuelve el número correspondiente a su código ASCII.

cite$: Para un elemento en la lista de datos, devuelve la clave interna que fue argumento en el comando "`\cite`" que provocó la inclusión de dicho elemento en la lista de datos. Sólo tiene sentido usarlo en comandos ITERATE o REVERSE.

duplicate$: Recibe una cadena y devuelve dos copias de la misma.

empty$: Recibe un dato y devuelve un "1" si ese dato consiste en un campo omitido o una cadena vacía, o un "0" en caso contrario.

Esta función suele usarse en todas las funciones de formateo de campos, que empiezan por comprobar que el valor del campo exista. Por ejemplo:

```
FUNCTION {format.title}
{ title empty$
```

```
    {""}
    { title }
  if$
}
```

Esta función empieza comprobando si el campo title está vacío u omitido, en cuyo caso envía a la pila una cadena vacía. En caso contrario envía a la pila el contenido del campo.

format.name$: Recibe una lista de nombres de personas, un número representativo de qué nombre hay que extraer de dicha lista, y una cadena representativa de cómo hay que formatear dicho nombre, y devuelve el nombre formateado. Al respecto debe tenerse en cuenta que la cadena de formateo no es usada para descomponer el nombre en sus diferentes partes, sino que esta descomposición es realizada automáticamente por BIBTEX de acuerdo con las reglas que se explicaron en la sección 2.2.3, pág. 16.

Una vez descompuesto el nombre, se aplica la cadena de formateo, que a su vez consta de varias subcadenas encerradas entre llaves. Cada una de las subcadenas representa una de las partes en las que BIBTEX descompone los nombres de personas. Para cada una de las subcadenas que componen la cadena de formateo se siguen las siguientes reglas:

1. Para indicar a cuál de las partes del nombre se refiere esa subcadena se usa, duplicada, la inicial de la denominación que en inglés tiene dicha parte del nombre, es decir: "{ff}" representa el nombre de pila (de *First*); "{vv}" representa la partícula (de *von*); "{ll}" representa los apellidos (de *Last*), y "{jj}" representa la partícula "Jr." presente en numerosos nombres anglosajones.
2. Si la inicial correspondiente a una de las partes del nombre no se duplica, significa que queremos que no se escriba esa parte completa, sino exclusivamente sus iniciales.
3. Todo texto que dentro de las llaves acompañe a la parte correspondiente del nombre, se imprimirá siempre y cuando en el nombre en cuestión dicha parte esté presente. Así, por ejemplo "`{, ff}`" imprimirá el nombre de pila separándolo del anterior elemento mediante una coma y un espacio en blanco. Pero en los nombres en los que no haya nombre de pila, no se imprimirá ni el nombre, ni la coma, ni el espacio en blanco.
4. Una ligadura colocada tras las iniciales representativas de una parte del nombre se interpreta como una sugerencia a BIBTEX para que si dicha parte del nombre consta de más de una palabra, no coloque las distintas palabras en líneas distintas. Pero es sólo una sugerencia. Si queremos forzar a BIBTEX para que haga eso, hay que escribir dos ligaduras seguidas.

5. Cualquier texto que en la cadena de formateo aparezca fuera de las subcadenas correspondientes a las distintas partes del nombre, se imprimirá literalmente en la cadena formateada.

Por ejemplo: Para que los nombres se impriman en el formato

`Apellidos, NombrePropio Partícula`

necesitaríamos la siguiente cadena de formateo:

`"{ll}{, ff}{vv}"`

y si quisiéramos que el nombre propio se abreviara a sólo las iniciales, deberíamos usar:

`"{ll}{, f}{vv}"`

Si queremos que en la cadena se incluya un texto, además del nombre, basta con incluirlo fuera de las llaves: por ejemplo la siguiente cadena:

`"Apellidos: {ll}, Nombre: {ff}"`

Aplicada a, por ejemplo, mi nombre, devolvería:

`Apellidos: Ataz López, Nombre: Joaquín`

Por defecto BIBTEX añade espacios en blanco o ligaduras entre las distintas partes de la cadena de formato, así como un punto al final de la misma y un punto y un espacio tras una inicial. Para indicar que esos caracteres no se añadan debemos usar unas llaves vacías tras la parte del nombre correspondiente. Y así, mientras "`{f}`" aplicado a "Joaquín" devuelve "J. ", "`{f{}}`" devolverá "J".

global.max$: Devuelve la longitud máxima permitida para cadenas. Se usa para evitar que la concatenación de cadenas de texto devuelva una cadena excesivamente larga. Esta longitud es definida internamente y en las versiones actuales de BIBTEX supera los 5000 caracteres.

int.to.char$: Recibe un dato numérico y devuelve el carácter ASCII cuyo código se corresponda con el número recibido. El dato numérico debe encontrarse entre 0 y 127.

int.to.str$: Recibe un número y devuelve una cadena de texto cuyo contenido sea dicho número. Por ejemplo si se recibe el número 67, se devolvería la cadena "67".

missing$: Recibe un dato y devuelve "1" si se trata de un campo omitido, y "0" en caso contrario.

newline$: Vuelca sobre el fichero «`.bbl`» el contenido del buffer de salida. Si el buffer de salida estaba vacío, entonces incluye en él un salto de línea. Dado

que la función «`write$`» produce saltos de línea razonables, sólo se recomienda esta función cuando por alguna razón se desea una línea en blanco o un salto de línea explícito.

num.names$: Recibe una cadena de texto y devuelve el número de nombres que hay en dicha cadena, es decir: el número de apariciones de la palabra "and" + 1.

pop$: Extrae el último dato de la pila, pero no lo imprime. Sirve simplemente para eliminar dicho dato de la pila.

preamble$: Devuelve una cadena que es la concatenación de todos los elementos «`@preamble`» presentes en el fichero «`.bib`».

purify$: Recibe una cadena de texto, realiza en ella ciertas transformaciones y devuelve la cadena resultante.

En particular, esta función preserva las letras, los números y los espacios. Reemplaza las tabulaciones, guiones y ligaduras con espacios y elimina todos los caracteres restantes (en concreto los que aparecen en el apéndice "C" de [5]. Los caracteres especiales son una excepción[60]. Los comandos de LaTeX, espacios en blanco, tabulaciones, guiones y tildes que se encuentren dentro de un carácter especial, son eliminados, pero los números y letras se respetan.

Esta función se usa para limpiar cadenas de texto antes de compararlas entre sí en las tareas de ordenación de la lista de referencias. Desde este punto de vista los textos "`\'e`" y "é" son diferentes, porque el primero se convertirá en "e" y el segundo no será modificado, seguirán siendo "é", pero en el momento de ordenar ambos textos se atenderá al código ASCII y "é" se colocará detrás, no sólo de "e", sino también de "z". Esta es una de las razones por las que en los ficheros «`.bib`» conviene usar el procedimiento estándar de LaTeX para representar caracteres acentuados.

quote$: Devuelve el carácter de dobles comillas.

skip$: No hace nada.

stack$: Extrae e imprime todo el contenido de la pila. Sirve para depurar ficheros de estilo mientras los estamos construyendo. La impresión se dirige hacia la salida estándar, no hacia el fichero de salida («`.bbl`»).

---

[60] En el contenido de los campos se considera *carácter especial* a un conjunto de caracteres encerrados entre llaves siempre y cuando las llaves no estén anidadas dentro de otras llaves (sin contar las posibles llaves de delimitación del campo en sí mismo considerado), y el primer carácter tras la apertura de las llaves sea "\".

substring$: Recibe una cadena de texto y dos números. El primero indica la posición de inicio y el segundo la longitud. Devuelve una cadena que se corresponde con la parte de la primera cadena que empieza en *inicio* y tiene la longitud especificada. Por ejemplo “Mesa”, “2”, “1” devolvería “e”, y “ordenador”, “5”, “4” devolvería “nado”. Si *longitud* es negativa, se cuenta hacia atrás, y así “ordenador”, “5”, “-4” devolvería “rden”. A efectos de esta función las llaves y caracteres especiales son contados como caracteres normales, y así, por ejemplo “`{\LaTeX}`”, “2”, “3”, devolvería “`\La`”.

swap$: Intercambia la posición de los dos últimos datos de la pila.

text.length$: Recibe una cadena y devuelve su longitud tal y como sería impresa, es decir: los caracteres especiales se cuentan como un sólo carácter y las llaves no se cuentan.

text.prefix$: Recibe una cadena y un número “n”, y devuelve una subcadena igual a los “n” primeros caracteres de la cadena recibida. Equivale en gran medida a «`substring$`», con la diferencia de que los caracteres especiales aquí son contados como un solo carácter y se ignoran las llaves.

top$: Extrae de la pila e imprime en la salida estándar y en el fichero log el último dato de la pila. No lo imprime en el fichero de salida «`.bbl`». Es útil para el depurado.

type$: Devuelve el nombre del tipo de registro actual (book, article, etc) o una cadena vacía si se trata de un tipo desconocido.

warning$: Recibe una cadena y la imprime en la salida estándar y en el fichero log, precedida de un mensaje de advertencia. Asimismo incrementa un contador dedicado al número de mensajes de advertencia generados.

width$: Recibe una cadena y devuelve un número representativo de su anchura en centésimas de punto, asumiendo para el punto la longitud que tiene en la fuente *cmr*10 de junio de 1987. Esta función es usada para comparar la anchura de las etiquetas y poder determinar el parámetro que hay que pasar al entorno «`thebibliography`».

write$: Recibe una cadena y la envía al buffer de salida que, a su vez, la enviará al fichero «`.bbl`».

El programador que invoque a una de estas funciones debe asegurarse de que la pila contenga los datos requeridos por la misma. Cuando una función espera que alguno de los datos recibidos a través de la pila deba ser de cierto tipo (numérico, alfanumérico, nombre de variable o función) y el dato extraido resulta no ser del tipo correcto, la función devolverá un “0” o una cadena vacía, dependiendo de que se trate de una función que devuelve un número o una cadena. Si se trata de una función que no devuelve nada, al recibir el dato incorrecto la función terminará su ejecución sin hacer nada.

8. Conclusión: Otros usos de BIBTEX

BIBTEX fue diseñado para la gestión de la bibliografía. Pero la flexibilidad de formato de los ficheros «`.bib`», y la potencia de los ficheros de estilo, permiten extender sus posibilidades hasta el punto de que podamos usarlo para muchas otras tareas. Podemos considerar a BIBTEX como una herramienta que nos permite insertar en un documento información procedente de una base de datos externa, así como decidir de qué manera debe formatearse tal información.

En [3] se explica cómo usar BIBTEX para generar una base de datos de problemas matemáticos. Y en [8] se habla del posible uso de BIBTEX para generar una lista de publicaciones y una libreta de direcciones. Díez de Arriba y Javier Bezos, por su parte, han diseñado el paquete Gloss que usa BIBTEX para generar glosarios, mientras que yo, que me dedico a la enseñanza del Derecho, he usado BIBTEX para con una base de datos de exámenes tipo test, y, en publicaciones de tipo jurídico, con una base de datos de sentencias, de tal modo que un simple comando "`\cite`" pudiera insertar en el texto un rótulo identificativo de una sentencia (del tipo «STS 21-4-56») y, al final del documento, una lista con las sentencias citadas.

Estos usos *alternativos* de BIBTEX pueden requerir un *paquete* para implementarlos. Sobre todo si se pretende que su uso no impida, en el mismo documento, el uso de BIBTEX al modo *normal*. Así ocurre en el caso de la generación de glosarios. O pueden exigir el diseño de un tipo nuevo de registro y una función que lo formatee incorporada a alguno de los estilos estándar de BIBTEX (como hice para la base de datos de sentencias y para la de exámenes tipo test), o simplemente un uso determinado de los registros normales.

Lo importante es que tengamos claro que BIBTEX esconde muchas más posibilidades de lo que a primera vista pudiera parecer.

Apéndice A: GNU Free Documentation License (Licencia GNU para documentación)

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom

to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of

formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires

Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations

of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Referencias

[1] Ataz López, Joaquín: *Creación de ficheros LaTeX con GNU Emacs*, 2004.
URL http://www.ctan.org/tex-archive/info/spanish/guia-atx/

[2] Ataz López, Joaquín: *Una introducción rápida a GNU Emacs*, 2004.
URL http://es.tldp.org/Tutoriales/doc-tutorial-emacs/

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

[4] Cascales Salinas, Bernardo, Lucas Saorín, Pascual, Mira Ros, José Manuel, Pallarés Ruiz, Antonio y Sánchez-Pedreño Guillén, Salvador: *El libro de LaTeX*. Pearson-Prentice Hall, 2003.

[5] Knuth, Donald E.: *The TEX Book*. Addison-Wesley, 1986.

[6] Kopka, Helmut y Daly, Patrick W.: *Guide to LaTeX*. Addison-Wesley, 4ª ed$^{\text{ón}}$., 2004.

[7] Lamport, Leslie: *LaTeX: A document preparation system*. Addison-Wesley, 1986.

[8] Markey, Nicolas: *Tame the BeaST*, 2005.
URL http://www.ctan.org/tex-archive/info/bibtex/tamethebeast/

[9] Mira Ros, José Manuel: «Bibliografía flexible: el sistema flexbib». En *TeXemplares*, , nº 6, págs. 8–26, 2004.
URL w3.mecanica.upm.es/CervanTeX/texemplares6.pdf

[10] Patashnik, Oren: *BibTeXing*, February 1988.
URL http://www.ctan.org/tex-archive/biblio/bibtex/contrib/doc/btxdoc.pdf

[11] Patashnik, Oren: *Designing BibTeX Styles*, February 1988.
URL http://www.ctan.org/tex-archive/biblio/bibtex/contrib/doc/btxhak.pdf

[12] Seidel, Luis: *Bases de datos bibliográficas, LaTeX y el idioma español*, March 1998.
URL ftp://tex.unirioja.es/pub/tex/doc/bibliogr.pdf

[13] Young, David: *Using BibTeX*, May 2002.
URL wwwmaths.anu.edu.au/~chrisw/LaTeX/bibtex.pdf