TEXemplares

Boletín de CervanTEX, Grupo de Usuarios de TEX Hispanohablantes

Año 3.º, número 2, primer trimestre 2001

TEXemplares

TEXemplares es el boletín de CervanTEX, el grupo de usuarios de TEX hispanohablantes. Ni *TEXemplares* ni CervanTEX comparten necesariamente la opinión de los colaboradores. Nada en *TEXemplares* debe entenderse como una postura oficial del grupo.

Las colaboraciones deben ser creadas con la clase `TeXemplar.cls`, disponible por CTAN en una versión preliminar. Se debe usar, siempre que sea posible, caracteres de 7 bit y acentuar con el método de abreviaciones de babel. Las figuras externas en `eps` se deben crear a 600 pt.

Este ejemplar se creo con las siguientes aplicaciones: `TeX, Version 3.14159`, `LaTeX2e <1997/12/01>` y `OzTeX 3.1.1`. Se imprimió a 600 pt. con una ampliación de 1,414 y luego reducido a su tamaño real en las copias.

Notación. Por necesidades de composición y contrariamente al comportamiento normal de `\verb`, el código puede ser dividido a final de línea. Las divisiones en un símbolo no se indican mientras que aquellas entre letras se hace con un guión estilístico. Eso quiere decir que un guión a final de línea *nunca es parte del código* mientras que los que haya al comienzo *sí lo son*.

Redacción. Javier Bezos, Eugenio Degroote, Pascual Lucas, Enrique Meléndez, Luis Seidel .

Depósito Legal y otros registros en trámite.

Editorial

Estimado lector,

Este segundo número, y el tercero que le seguirá, corresponden a la edición de las ponencias presentadas en el primer encuentro del grupo de usuarios de TEX hispanohablantes, el EGUTH'99 que se celebró en Madrid los días 13 y 14 de septiembre de 1999. Puede decirse que el encuentro tuvo un éxito indiscutible, gracias a la inmejorable organización del Instituto de Ciencias de la Educación (ICE) de la Universidad Politécnica de Madrid y al dedicado y desinteresado trabajo de Luis Seidel, quien aceptó sobre sus hombros la ingrata tarea de pedir y editar las ponencias,

Por distintas razones este segundo número aparece casi un año después de que lo hiciera el primero. Ha sido posible gracias a la labor de los editores, Javier Bezos, Pascual Lucas, Luis Seidel y Enrique Meléndez, quienes han dedicado mucho tiempo a que la calidad en su edición sea la adecuada. Eugenio Degroote se apuntó a esta tarea de edición y debe contarse entre los editores, aunque su participación en este número se ha visto limitada por motivos de trabajo personal.

Los artículos que se presentan en este número corresponden, com se ha dicho, a seis de las ponencias del EGUTH'99. En la primera, Jesús Carretero y Santiago Rodríguez nos cuentran su trabajo para desarrollar un corrector ortográfico en castellano usando las utilidades del programa ispell. Para ello han desarrollado la especificación formal de las reglas de derivación y el etiquetado de las palabras que componen el diccionario. Gabriel Valiente, *alma mater* de Tirant lo TEX describe cómo contribuye , según su experiencia, el uso y formación en los sistemas TEX y LATEX al desarrollo curricular en materias científico-técnicas. Las tecnologías ligadas a Internet son el objeto del tercer artículo, cuyo autor es Luis Seidel, donde se

pinta un amplio panorama de uso de TEX para la edición orientada a medios electrónicos. Otro corrector ortográfico ocupa el artículo de Juan Luis Varona. Éste está pensado para ser usado con varios editores y correctores como por ejemplo Excalibur para Macintosh, WinEdt, para Windows y amSpell (MS-DOS). Javier Sanguino hace una excursión por varios de los paquetes usados frecuentemente en la generación de documentos y que facilitan la edición de textos complejos. Finalmente, Enrique Meléndez y Roberto Herrero explican el sistema WEB, que no tiene nada que ver con la red, sino que es un sistema de documentación de un programa usando TEX y combinando el código fuente con las explicaciones.

Merece especial mención Javier Bezos, quien ha contribuido de manera muy especial a la edición del boletín de CervanTEX, al hacer disponible el estilo y las utilidades para ello (además de su impagable labor en el desarrollo del estilo spanish para babel) y contribuyendo con su buen hacer tipográfico a la calidad del producto.

El agradecimiento va también a los autores, que han aportado lo esencial del número, sus contribuciones. Se quiere animar desde aquí a todos los usuarios de TEX y amigos de TEXemplares a mantener vivo este boletín remitiendo contribuciones para próximos números.

TEXemplares 2/2001, págs. 5-15

Corrector ortográfico de libre distribución basado en reglas de derivación

Jesús Carretero Santiago Rodríguez
Facultad de Informática
Universidad Politécnica de Madrid, 28660 Madrid, España
e-mail: {srodri, jcarrete}@fi.upm.es

Resumen

La carencia de herramientas software de libre distribución que permitan la corrección de textos escritos en castellano llevó a los autores a la construcción de un diccionario de castellano basado en el programa de libre distribución *ispell*. Este artículo presenta la integración del diccionario al entorno de la herramienta *ispell*, haciendo especial hincapié en los aspectos de la especificación formal de las reglas de derivación, etiquetado de las palabras que componen el diccionario raíz y la generación final del diccionario. El diccionario se distribuye como una herramienta de libre distribución desde 1994 bajo los términos de la *General Public License* de *Free Software Foundation*.

Palabras Clave: Lenguaje Natural, Especificación Formal, Software de Libre distribución, LaTeX.

1. Introducción

La introducción de los computadores en el procesamiento de textos ha demostrado la carencia de herramientas especializadas (correctores ortográficos, correctores gramaticales, etc.) para el castellano. Ésta es la tercera lengua más extendida y, sin embargo, la disponibilidad de este tipo de herramientas está muy lejos de otras lenguas mucho menos extendidas, pero con mayor influencia tecnológica que los países de habla hispana.

La importancia potencial del castellano en un futuro próximo llevó a los autores a desarrollar algunas herramientas gramaticales *software*. Además, para promocionar la utilización de estos programas, se pensó en su distribución totalmente gratuita. Una de estas herramientas es el diccionario de castellano para el corrector ortográfico *ispell* desarrollado por Geoff Kuenning que permite la incorporación de distintos diccionarios (Ispell se puede obtener mediante ftp anónimo del servidor `ftp.math.orst.edu` en `/pub/ispell-3.1/ispell-3.1.20.tar.gz`).

Uno de los principales objetivos impuestos en el desarrollo del diccionario de castellano fue su distribución totalmente gratuita para permitir su utilización al mayor número de usuarios posible. Por otra parte el diccionario debe ser exhaustivo, es decir, debe contener el mayor número posible de palabras aceptadas en castellano, así como la mayor parte de sus derivaciones. Puesto que es una herramienta de libre distribución debe ser fácil de mantener ya que se espera la colaboración de los usuarios finales para actualizar y mejorar tanto el diccionario raíz como el conjunto de reglas de derivación. Por último, debido a la diversidad de vocabulario del castellano, dependiendo de la zona geográfica del usuario que utilice la herramienta se utiliza un conjunto de palabras ligeramente distinto del resto. Por tanto el proceso de generación del diccionario debe permitir al usuario seleccionar el conjunto de palabras que mejor se adapte al que se utiliza en su zona geográfica.

El principal problema encontrado en el desarrollo del diccionario fue la adaptación de las reglas gramaticales castellanas a una especificación formal. A diferencia del inglés, el castellano contiene un número muy elevado y complejo de reglas de derivación a partir de una palabra raíz. Las principales tareas que se realizaron fue la formalización de las reglas de derivación gramaticales y la generación de un conjunto de palabras etiquetadas (palabras con su conjunto de reglas a aplicar). El desarrollo de esta herramienta se inició a comienzos de 1994. El primer prototipo estuvo finalizado a mediados de 1994 y se dedicó a su uso interno para detectar errores. Se distribuye de forma gratuita desde finales de 1994[1]. Actualmente se distribuye la versión 1.6 (Abril 1999).

[1]Se puede obtener mediante ftp anónimo del servidor `ftp.fi.upm.es` en `pub/unix/espa~nol.tar.gz` o mediante el URL `http://www.datsi.fi.upm.es/~coes/`

2. Características morfológicas del castellano

El castellano es una lengua que derivó del latín y tiene una gramática compleja. Para llevar a cabo la construcción de cualquier plataforma léxica la primera tarea que se debe llevar a cabo es el estudio de la gramática castellana [1]. Este estudio debe permitir formalizar el conjunto mínimo de reglas necesario que permita extraer el conjunto de palabras reconocidas por la lengua castellana a partir de un conjunto de palabras raíces mínimo. Para alcanzar dicho objetivo se construyó un árbol de derivación a partir de una palabra raíz. Una versión simplificada de dicho árbol se muestra en la figura 1.

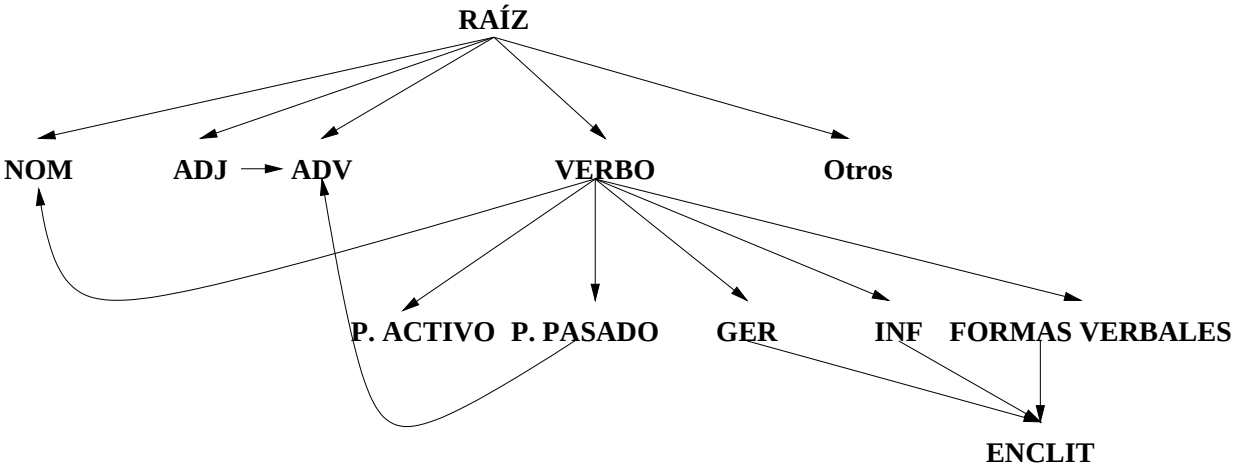


Figura 1: Estructura Morfológica simplificada del castellano

Los principales problemas que se encontraron en la construcción de este conjunto de reglas de derivación vinieron originados por las características del castellano. Los más relevantes se exponen a continuación:

Derivaciones de género y número. Los adjetivos (ADJ) y sustantivos (NOM) tienen género (masculino o femenino) y número (singular y plural). La situación habitual es que un adjetivo o nombre tenga derivación tanto en género como en número. Por ejemplo: *perro* → *perra* y (*perros, perras*). Otros casos únicamente tienen un género y, por tanto sólo admiten derivación en número. Es el caso de un sustantivo masculino como *álamo, álamos*, o femenino como *casa, casas*.

Conjugación verbal. Cada una de las tres conjugaciones verbales del castellano tienen 40 derivaciones temporales (P. ACTIVO, P. PASADO,

GER, INF y FORMAS VERBALES). Como es sabido, los verbos regulares tienen un conjunto estricto de reglas de derivación que son idénticas para todos los de una misma conjugación. Los verbos irregulares tienen al menos una derivación diferente que las derivaciones regulares correspondientes a su conjugación. Las derivaciones irregulares se agrupan en alrededor de 100 tipos diferentes de irregularidades [1].

Formas enclíticas. Algunas derivaciones verbales se generan añadiendo una forma pronominal al final de una forma verbal (ENCLIT). En el castellano escrito se pueden encontrar dos formas enclíticas diferentes: los verbos pronominales, cuyas formas enclíticas se generan añadiendo los sufijos *-me, -te, -se, -nos* y *-os* en el infinitivo y en el gerundio (*amar* → *amarte*), y los verbos transitivos, cuyas formas pronominales se generan añadiendo las terminaciones *-lo, -la, -los, -las, -le* y *-les* (*amar* → *amarla*). Ambas formas enclíticas se pueden combinar para formar enclíticos más complejos (*ajustar* → *ajustármelo*). Esto genera un conjunto de reglas de complejidad $O(n^2)$. Además, estas formas enclíticas se ven afectadas por las irregularidades que presentan algunos verbos en su gerundio (*vestir* → *vistiéndote*), lo que incrementa el grado de complejidad para las formas enclíticas.

Nombres derivados de verbos. Algunos sustantivos son formas derivadas de un verbo como *imaginar* → *imaginación* o *abatir* → *abatimiento* (VERBO → NOM).

Adverbios derivados de adjetivos. Gran parte de los adverbios modales se generan añadiendo el sufijo *-mente* a un adjetivo (ADJ → ADV): *tranquilo* → *tranquilamente*.

Superlativos y diminutivos. Las formas regulares de superlativos se forman añadiendo el sufijo *-ísimo* a un adjetivo (*grande* → *grandísimo*). Los diminutivos se forman añadiendo los sufijos *-ico, -ito* y *-illo* a un adjetivo o nombre.

Vocales acentuadas. Hay muchas particularidades relacionadas con las derivaciones en género y número que se han tenido en cuenta al hacer el estudio del modelo. Algunas palabras pierden una vocal acentuada sustituyéndola por su equivalente no acentuada: *gañán, gañanes*.

Teniendo en cuenta las características descritas en los párrafos anteriores se ha desarrollado un conjunto de reglas formales que comprende un

extenso subconjunto de las que conforman la gramática castellana. Cada una de las entradas del diccionario que contiene las palabras raíces tiene una etiqueta que representa una lista de reglas de derivación que se deben aplicar a dicha palabra para obtener sus formas derivadas.

3. Implementación del modelo.

Las características gramaticales estudiadas en el apartado anterior han llevado a la realización del modelo ajustándose a las restricciones que imponía la herramienta *ispell*. Estas restricciones se basan en agrupar un conjunto de reglas (clase) al que se asocia una etiqueta que será referenciada en las etiquetas del diccionario raíz.

Puesto que el uso del castellano se basa en gran parte en las formas derivadas (un verbo castellano tiene 55 formas derivadas), las reglas de derivación del diccionario incluyen todas las derivaciones de los verbos regulares. Además, se han incorporado reglas adicionales para tener en cuenta la mayor parte de los patrones por los que se rigen las derivaciones de los verbos irregulares. La inclusión de las formas derivadas de los verbos *ser*, *ir*, *haber* y *estar* se han incluido en su totalidad en el diccionario raíz al no adecuarse fácilmente a ninguno de los patrones considerados. En resumen, el conjunto de reglas implantado para especificar la gramática castellana contiene alrededor de 3.300 reglas agrupadas en 57 macroreglas o clases.

Cada macroregla que se describe a continuación refleja un aspecto particular de la gramática castellana que se ha descrito en la sección anterior. Cada una de las reglas de derivación que componen una clase o macroregla trata un caso particular [2, 3, 4]. La condición que se muestra en la primera columna de cada uno de los ejemplos representa la aceptación de una palabra para ejecutar la acción que se muestra en la segunda columna. Si una palabra termina con el sufijo especificado, se realiza la acción subsiguiente. Esta acción se basa en sustituir un morfema de la palabra raíz por otro, o simplemente añadir un morfema adicional.

Derivaciones de género y número. Se han incluido dos macroreglas que realizan estas derivaciones. Las derivaciones en número incluyen 11 reglas. La regla a aplicar depende de la terminación de la palabra raíz sobre la que se aplica la regla. La macroregla de derivación en género y número se compone de 20 reglas. A continuación se muestran algunos ejemplos de derivación de estas clases:

Derivaciones en número			Derivaciones en género y número		
Condición	Acción	Ejemplo	Condición	Acción	Ejemplo
[AEIOU]	S	vacas	O	-O, A	amiga
Z	-Z, CES	arroces	O	S	amigos
ÚN	-ÚN,UNES	atunes	[^AONS]	ES	pastores

La última regla del ejemplo anterior se muestra la generación del plural masculino y femenino para aquellas palabras que no acaban en *a*, *o*, *n* ni *s*.

Conjugación Verbal. Las reglas de derivación que permiten generar las formas verbales se agrupan en cuatro clases: dos de ellas se aplican a verbos regulares y las otras dos a verbos irregulares. Alrededor de 200 reglas componen las dos clases que derivan los verbos regulares mientras que el conjunto de derivaciones de los verbos irregulares se compone de unas 2.500 reglas. En este último aspecto es donde se ha dedicado la mayor parte del esfuerzo. Sin embargo su formalización ha sido factible puesto que las formas irregulares del castellano siguen patrones de derivación bien definidos: *-ontar* → *-uento*, *-oder* → *-uedo*, *-ervir* → *-irvo*, etc. Algunas reglas de derivación para verbos regulares e irregulares se muestran a continuación:

Verbos Regulares			Verbos Irregulares		
Condición	Acción	Ejemplo	Condición	Acción	Ejemplo
AR	-AR, O	amo	IAR	-IAR, ÍO	envío
CER	-CER, ZO	venzo	OÑAR	-OÑAR, UEÑO	sueño
CIR	-CIR, ZO	zurzo	SABER	-ABER, É	sé

Estas reglas no han tenido en cuenta los verbos *ser, estar, ir* y *haber* puesto que no existen un conjunto de patrones que permitan derivar todas sus formas verbales a partir del infinitivo. Todas sus formas derivadas se han incluido explícitamente en el diccionario de palabras raíces.

Formas enclíticas. Los verbos regulares incluyen alrededor de 200 reglas de derivación para generar las formas enclíticas, mientras que los verbos irregulares incorporan en torno a 400. Estas reglas representan los

enclíticos generados por las derivaciones pronominales, transitivas y combinadas de ambas. Estas reglas únicamente se aplican a las formas del gerundio e infinitivo.

Nombres derivados de verbos. Se han tenido en cuenta los nombres acabados en *-miento* y *-ción* que se derivan de verbos a partir de dos macroreglas.

Adverbios derivados de adjetivos. Se ha considerado una macroregla (clase) que genera los adverbios terminados en *-mente*.

Superlativos y diminutivos. Actualmente únicamente los superlativos regulares se han considerado y constituyen una clase.

4. Generación del diccionario

El léxico para esta plataforma ha sido extraído de un *Corpus de Español* compilado por los autores. Este corpus, que contiene más de 20 millones de palabras, incluye textos extraídos de las siguientes fuentes: textos de periódicos españoles (ABC Cultural, El Mundo, El Periódico, etc.); libros seleccionados, como la Biblia; textos técnicos (informes técnicos, artículos, proyectos de fin de carrera, diccionarios técnicos y libros); Corpus oral [5]; Versión concisa del diccionario Español-Inglés Collins [6].

El *léxico básico* resultante contiene más de 80.000 palabras distintas, 53.000 de las cuales han sido ya etiquetadas de acuerdo a las reglas de derivación mostradas en la sección anterior. Las restantes 27.000 están en proceso de etiquetado. El etiquetado se hace de forma semiautomática mediante una herramienta que extrae los morfemas de cada palabra y propone una o varias *etiquetas tentativas*. Sin embargo, las formas derivadas son comprobadas manualmente para verificar la corrección o no del etiquetador. El *léxico de referencia* usado para desarrollar COES es el diccionario de la *Real Academia Española* (RAE), la institución oficial que vela por la pureza del lenguaje español y admite las nuevas palabras del mismo.

La versión de libre distribución de COES incluye un fichero de afijos y varios ficheros de léxico:

- `espa~nol.words` contiene una lista de palabras del diccionario oficial de español [7].

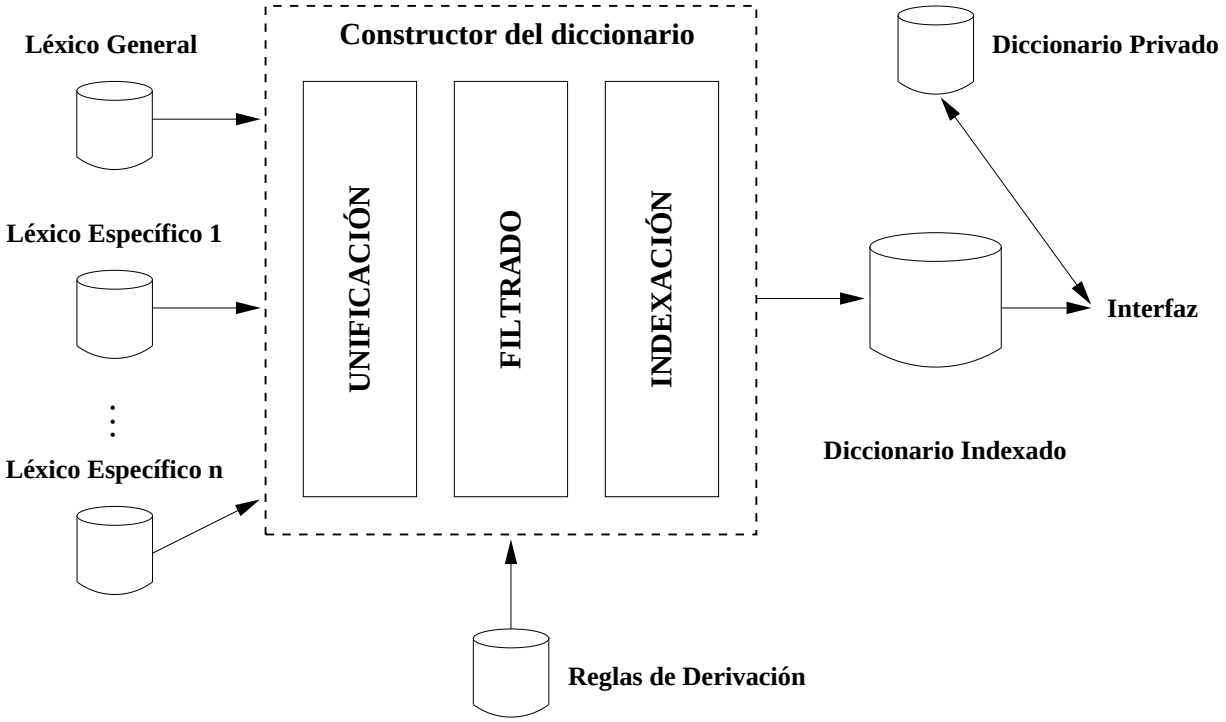


Figura 2: Generación del Diccionario

- `espa~nol.comp` contiene una lista de palabras que no aparecen en el diccionario oficial de la lengua española, pero de uso habitual en los textos técnicos.
- `antiguas.words` contiene una lista de palabras que aparecen en el diccionario oficial de español, pero etiquetadas como palabras en desuso.
- `espa~nol.nofl` contiene una lista de palabras que no aparecen en el diccionario oficial de español, pero que han sido frecuentemente encontradas en el corpus.
- `espa~nol.propios` contiene una lista de nombres propios.

Cuando se aplican las reglas de derivación al léxico básico usado en COES actualmente se crea un diccionario que contiene más de 650.000 palabras. El diccionario de español se construye usando la herramienta *ispell*, que aplica las reglas de derivación elaboradas por los autores, siguiendo el formato de esta herramienta, al léxico básico. *ispell* sigue cuatro pasos básicos para generar del diccionario (figura 2):

1. Generación de las reglas de derivación a partir del fichero de reglas.

2. Unificación de las entradas del diccionario para evitar redundancias y formas ilegales.

3. Interpretación de las reglas de derivación para calcular las formas derivadas.

4. Construcción de un árbol indexado para conseguir una búsqueda eficiente en el diccionario.

Existe la posibilidad de que los usuarios puedan generar diccionarios *particularizados* mezclando varios ficheros de léxico cuando se construye el diccionario. Esta opción permite a cada usuario incluir sus propios léxicos (que deberían estar etiquetados). Además, los usuarios pueden particularizar el diccionario eligiendo el formato en que se codificarán los caracteres especiales (ü, ñ y letras acentuadas), que no se encuentran definidos en el conjunto básico de caracteres ASCII de siete bits. Para permitir esta particularización, se proporciona a los usuarios la codificación de estos caracteres en los formatos más habituales cuando se definen las reglas.

Actualmente se proporcionan los siguientes formatos distintos:

latin1	Formato TeX	Formato LaTeX	Html
á	\' a	'a	á
é	\' e	'e	é
í	\' \i	'i	í
ó	\' o	'o	ó
ú	\' u	'u	ú
ñ	\' n	'n	ñ
ü	\ü	"u	ü
Á	\' A	'A	Á
É	\' E	'E	É
Í	\' \I	'I	Í
Ó	\' O	'O	Ó
Ú	\' U	'U	Ú
Ñ	\' N	'N	Ñ
Ü	\Ü	"U	Ü

Formato **msdos**: Las letras acentuadas se codifican utilizando el código ASCII MS-DOS extendido.

Para ejecutar el ispell con un determinado formato:

```
ispell -T <formato> -d español <fichero>
```

El diccionario se puede generar en cualquier sistema operativo para el que exista una versión de ispell. Esto incluye cualquier computador que ejecute alguna versión de Unix y, además los sistemas Windows NT y Windows 95/8.

5. Conclusiones y trabajo futuro

En este trabajo se ha presentado un diccionario de español desarrollado para la herramienta *ispell*, diccionario que está siendo usado por una comunidad creciente de usuarios. La versión actualmente existente de COES puede ser mejorada en los aspectos siguientes: Elaboración de diccionarios locales y temáticos, para dar cabida a palabras usadas en áreas restringidas de la comunidad hispanohablante o en entornos lingüísticos especiallizados (leyes, medicina, etc.); optimización de reglas; incrementar el léxico básico para reducir la tasa de error de COES y aumentar su eficiencia.

La difusión de este trabajo como una herramienta de libre distribución ha permitido una rápida extensión de la misma, encontrándose actualmente plenamente integrada con herramientas de libre distribución que usan ispell (por ejemplo *emacs*). Además se están manteniendo conversaciones con representantes del proyecto Lucas (Linux en Castellano) para mejorar la distribución y la calidad de COES.

El mantenimiento del diccionario y su depuración es una tarea que han asumido los autores casi en su totalidad. Sería interesante contar con la colaboración de los usuarios para detectar palabras erróneas, ausentes, reglas con erratas, etc., aunque la experiencia demuestra que los usuarios son poco colaboradores. En este sentido existe una dirección de correo a la que se pueden enviar cualquier tipo de sugerencia o error detectado:

`espanol-bugs@datsi.fi.upm.es`

Actualmente están en desarrollo algunas nuevas utilidades para COES. Un *tesauro*, que usará intensivamente las reglas de derivación, estará disponible pronto. Además se está llevando a cabo un estudio preliminar de las reglas y modelos sintácticos y gramaticales del español [8, 9] con el propósito de construir un corrector sintáctico en un futuro próximo.

Referencias

[1] R. A. E. de la Lengua. *Esbozo de una Nueva Gramática de la Lengua Española*. Espasa Calpe (1991).

[2] S. Rodríguez y J. Carretero. Building a Spanish speller. En *Taller sobre Software de Libre Distribución*. Universidad Carlos III, Madrid, España (1995).

[3] S. Rodríguez y J. Carretero. A formal approach to Spanish morphology: the COES tools. En *SEPLN'96 Conference Proceedings*, páginas 118–126. Sevilla, España (1996).

[4] J. Carretero y S. Rodríguez. Building lexical tools to manage information written in Spanish. *Journal of Information Science*, 22:391–399 (1996).

[5] F. Marcos, A. Ballester, C. Santamaría, E. Pertierra, O. Brandeo, y P. Díez. Corpus oral de referencia de la lengua española contemporánea. informe técnico, Universidad Autónoma de Madrid (1992).

[6] C. Smith. *Collins English-Spanish Dictionary*. Collins (1988).

[7] R. A. E. de la Lengua. *Diccionario de la Lengua Española*. Espasa Calpe, 21.ª edición (1992).

[8] J.Hallebeek. *Morfología y Sintaxis del Español: Introducción al Análisis Oracional*. Playor, Madrid, España (1994).

[9] E. Tzoukermann y M. Liberman. A finite-state morphological processor for Spanish. En *Proceedings of the 13th International Conference on Computational Linguistics (COLING 90)*, páginas 277–281 (1990).

TEXemplares 2/2001, págs. 23-31

TEX en la Web: Publicación electrónica de documentos científicos

Luis Seidel Dpto. de Física Aplicada
ETSI Industriales. UPM
seidel@faii.upm.es

1. En el principio...

Hace muchos, muchos años, en una galaxia lejana, D. E. Knuth concibió un sistema para componer tipográficamente documentos con un gran contenido matemático. Poco después, en la misma galaxia, Leslie Lamport puso al alcance de casi todos dominar la potencia de TEX. Lo que empezó como un *divertimento* de *computer scientists* dio origen a un lenguaje universal para el intercambio de documentos científicos, al menos entre matemáticos, físicos, ingenieros, informáticos, economistas,...

Durante los primeros 25 años de lo que ahora llamamos INTERNET, científicos de tales áreas de conocimiento han escrito artículos, libros, Tesis... y los han intercambiado por correo electrónico. Desde hace al menos diez años esos documentos podían ir escritos en LATEX con la confianza de que el receptor del mensaje electrónico podría obtener una versión impresa y correctamente tipografiada igual al documento original. Cuando los procesadores de texto ahora en boga estaban en pañales, el proceso descrito era suficientemente estable como para que muy respetables revistas científicas admitieran originales enviados por correo electrónico y escritos en LATEX[2].

[2]En lo que se refiere al campo de la Física, la American Physical Society admite *compuscritos* preparados por el autor en REVTEX para su publicación en *Physical Review* desde 1987.

El proceso de publicación se simplifica y se acorta enormemente. Otra ventaja de esta solución es la posibilidad de difundir inmediatamente entre colegas un trabajo enviado para su publicación que no estará disponible hasta dentro de varios meses (*preprints*). Si los documentos se escriben en LaTeX, tambien es factible crear un depósito automatizado y de fácil acceso para un tema de interés[3]. Cualquiera de estos caminos acaba en un documento impreso, de forma que sólo la transmisión es electrónica, no tanto la publicación.

En una galaxia distinta, y en tiempos mucho más recientes, se ha producido el auge de la informática de consumo e INTERNET se ha convertido en un fenómeno de masas y terreno codiciado por multinacionales y grupos mediáticos. Palabras como *Web*, *ciberespacio* o *multimedia* parece que deban aparecer en una conversación casual con la misma naturalidad que *fichaje histórico, boda del siglo* o *anticiclón de las Azores.*

Esta situación es comparable a la del buen conocedor que sabe de la existencia de una playa recóndita y desierta en la costa de Galicia, de dificil acceso, a la que acude con sus amigos a hacer *surf*. Un día descubre que han hecho una autopista hasta la arena, y la playa está llena de bañistas. Escribo esta nota para los que piensan que algo se ha perdido con facilitar el acceso. Mi opinión es que muy pocos de los bañistas saben hacer *surf*. Y para los que ya sabemos, es muy fácil pasar al *wind-surf* [4].

Este artículo pretender resumir el panorama de soluciones para la publicación en INTERNET de documentos científicos y dejar claro el manejo de dos de esas soluciones, accesibles a la mayoría. El tema es muy extenso y para más —y mejor— información se puede acudir al *LaTeX Web Companion* [1]. Tras leer y aprender a manejar los paquetes descritos, estará claro que la salud y el porvenir de TeX en los tiempos de INTERNET son óptimos.

Desde el punto de vista del usuario (y esto se publica en el boletín de un Grupo de Usuarios) no importa tanto la descrición histórica, técnica o filosófica. Uno quiere saber como se maneja una herramienta, y sólo una, para realizar una función. Lo mejor es —siempre— leer el manual y aprender haciendo. Pretendo únicamente indicar el manual que a mí me ha servido y mostrar las soluciones que he encontrado.

Para una discusión de la publicación científica efectiva se puede ver `<http://www.cl.cam.ac.uk/~mgk25/publ-tips.html>` y también `<http:`

[3] En Los Alamos National Laboratory es bien conocido el servicio de *e-prints* `<http://xxx.lanl.gov>` que tiene una réplica en España `<http://xxx.unizar.es>`.

[4] La metáfora pone a los TeXeros de surfistas y compara el fenómeno de la Web a la aparición del *wind-surf*.

//hutchinson.belmont.ma.us/tth/webmath.htm>.

2. Soluciones para la publicación en la Web de documentos con contenido matemático

El avance en el tratamiento automatizado de la documentación va más allá de la mera tipografía. Desde la aparición del SGML (Standard Generalized Mark-up Language) en 1986 se pretendió que el archivo que describe el contenido de un documento contenga también información sobre su estructura: qué es un Capítulo, qué es un Título,... Para los usuarios de LaTeX esto es bien conocido. Una derivación del SGML fue el HTML (HyperText Mark-up Language), que añade el concepto de enlace o hipervínculo, que permite *navegar* o moverse por un documento de modo no lineal, o pasar a otros documentos (otro viejo conocido en LaTeX, a través de las referencias cruzadas, las citas, ...). Este lenguaje de *marcado* junto con la definición de URL (Uniform Resource Locator) y el uso de algunos protocolos de comunicación (como el HTTP, HyperText Transfer Protocol) permitieron el nacimiento de la *World Wide Web* allá por 1992 precisamente en un Centro de Investigación científica básica[5].

En el mundo de la impresión profesional, el estándar es PostScript (marca registrada de Adobe). Una evolución de este lenguaje de descripción de páginas hacia la publicación (difusión) electrónica se conoce como PDF (Portable Document Format) nacido en 1996. Con una orientación a objetos, sin las capacidades de programación de PostScript pero permitiendo que la composición tipográfica de un documento pueda respetarse en cualquier plataforma (pantalla o impresora), se ha impuesto rápidamente como un estándar *de facto* en la publicación de documentos que deben conservar su composición a través de distintas plataformas. La Web está llena de libros y documentos de todo tipo en PDF[6].

En el momento presente, los *gurús* anuncian las bondades del XML (eXtensible Mark-up Language), cuyo estándar ha sido definido en 1998 y que promete ser un feliz compromiso entre la generalidad del SGML y la facilidad de uso de HTML. Esto, junto a su conjunción del contenido y el contexto le presentan como la panacea, pero por si acaso, vamos a quedarnos en lo que tenemos hoy.

[5] El CERN, centro europeo de investigación nuclear.

[6] Que haya encontrado recientemente, desde el plano del metro de Valencia hasta las Encíclicas de Juan Pablo II.

Desde el punto de vista científico, si se necesitan publicar documentos con contenido matemático, el lenguaje de marcado debe contemplar este tipo de información como distinta del texto ordinario[7]. El consorcio W3C, que define los estándares en la WWW ha discutido largamente sobre la inclusión de fórmulas matemáticas. A dia de hoy aunque existe el estándar MathML, solo un navegador experimental (Amaya) permite visualizar documentos con ese marcado.

Por tanto, no hay todavía una solución única mientras se espera la llegada del estándar perfecto. Incluso soluciones muy profesionales como la revista científica *The New Journal of Physics* `<http://www.njp.org>` patrocinada por The Institute of Physics y la Sociedad Alemana de Física, publica sus artículos en HTML y PDF y admite originales en LaTeX[8]. Voy a describir a continuación dos soluciones para convertir LaTeX a HTML y PDF.

3. De LaTeX a HTML: LaTeX2HTML

Mientras WWW sea igual a navegador (IE ó Netscape) y navegador sea fundamentalmente visualizador de HTML (con añadidos o *plug-ins*) la primera solución que buscamos pasa por convertir TeX en HTML. Mencionaré por completitud que se pueden var documentos en TeX directamente dentro de un navegador con un *plug-in* de IBM que se llama TechExplorer. También un paquete llamado WebEQ utiliza *applets* de Java para representar las ecuaciones y hay al menos tres desarrollos para convertir LaTeX en HTML: TeX4ht, TTH y LaTeX2HTML.

Voy a describir el LaTeX2HTML de Nikos Drakos. Tiene un cierto respaldo *oficial* por ser el utilizado en la conversión de la documentación de LaTeX 2_ε a HTML desde la *release* de junio de 1999. Consta de unos cuantos programas escritos en PERL y se encuentra en la versión 99.2 (beta8, de diciembre de 2000). En la actualidad es mantenido por Ross Moore y cuenta con la colaboración de un importante número de personas. Se puede obtener en `<www.latex2html.org>`. Instrucciones detalladas para instalar LaTeX2HTML acompañando a MikTeX bajo Windows están en `<http://feynman.faii.etsii.upm.es/~seidel/l2h>`.

El soporte a distintos paquetes implica escribir otro programa en PERL que explique que hace ese paquete y como traducirlo a HTML. Por lo que

[7] Lo que ha hecho TeX de toda la vida ...

[8] Y en Microsoft Word !

nos interesa, el soporte para español[9] se reduce a la traducción de Chapter, Index,... Se entiende que el *input encoding* del español es el iso-latin1 y no reconoce las abreviaturas de spanish como `'a,~n,...`

Si las ecuaciones son medianamente complicadas, o incluyen símbolos que no estén en los tipos del navegador, las convierte en gráficos, para lo que necesita algunas aplicaciones adicionales (ghostscript,netpbm) además de PERL y LaTeX. Es recomendable compilar primero el documento en LaTeX (para tener los archivos `.aux, .toc,...`) y luego llamar a `latex2html`. Si el programa consigue finalizar con éxito (después de llenar la pantalla de puntos y símbolos y de gastar enormes cantidades de recursos) tendremos en una carpeta con el nombre del documento una colección de archivos `.html` y `.gif` que representar a nuestro documento original.

Algunas de las opciones que se añaden a la línea de órdenes y que son útiles son:

`-no_math -html_version 3.2,math` Estas dos opciones combinadas hacen que las fórmulas sean texto en lo posible y gráficos en lo demás. Da la mejor calidad.

`-split n` Especifica la menor unidad del documento (Capítulo, Sección, ...) que convierte en archivos HTML distintos. 3 indica el nivel de *Sección.*

`-antialias` Intenta suavizar los tipos para que sean más legibles en pantalla.

`-no_navigation` Suprime los botones que permiten la navegación por el documento.

`-info 0` Suprime la información sobre la conversión y el programa que la ha realizado que se incluye por omisión al final del documento.

4. De LaTeX a PDF: pdfTeX

Hoy, el formato idóneo para la publicación electrónica (en pantalla, o en la WWW) de documentos con una composición fijada por el autor es sin duda PDF: es un estándar, con el apoyo de la industria, portable, rico en posibilidades de hipertexto, de formularios y de contenido gráfico. Para

[9] Que debemos a Jesús M. González-Barahona.

ver o imprimir un documento PDF lo único que se necesita es el *Acrobat Reader* que es un programa de Adobe de distribución libre[10].

La producción de documentos en PDF estaba ligada inicialmente al *Distiller* de Adobe que convierte PostScript en PDF. Pero como el formato está bien documentado empiezan a aparecer programas con salida directa a PDF. Se puede encontrar información en<`http://www.purepdf.com`>. Dentro del mundo TEX aparecen una multitud de soluciones: algunas distribuiciones comerciales (como Y&Y) lo permiten. El paquete DVIPDFM (de Mark Wicks) permite el paso *directo* de DVI a PDF[11].

Ha aparecido recientemente un nuevo conjunto de macros llamado Con TeXt que utiliza extensamente (y con un enfoque moderno) las capacidades de hipertexto y de pdfTEX. Más información (con ejemplos espectaculares) en <`http://www.pragma-ade.nl`>.

La forma más directa de generar PDF es utilizar PDFTEX, una modificación de TEX escrita por Han The Thanh. Es una modificación en el sentido de que se añaden algunas primitivas nuevas. Se encuentra en fase *beta* en la versión 0.14b. Como se pueden crear archivos de formato al estilo de lo que hace iniTEX para LATEX, utilizar PDFTEX sobre un archivo LATEX es tan sencillo como[12]:

```
pdflatex archivo_latex.tex
```

Y con gran probabilidad obtendremos un bonito archivo PDF.

Uno de los motivos por los que, suponiendo que utilizamos las fuentes Computer Modern o AMS-fonts, el archivo se *ve bien* en Acrobat Reader es que se utilizan fuentes Postscript Type 1. Gracias a un consorcio de editoriales y casas de software, las fuentes Computer Modern y las AMS en versión Type 1 están disponibles (bajo el Copyright de la American Mathematical Society) gratuitamente. No puedo enfatizar suficientemente este punto: para que el documento escrito para fuentes CM o AMS sea visible *decentemente* en Acrobat Reader es *necesarioi* y *obligatorio* emplear fuentes Type 1 (vectoriales e independientes de la resolución). Hay que olvidar (al menos para este fin, para otros son muy válidas) las fuentes `.pk` generadas por METAFONT (Type 3 en la jerga de PostScript), *raster* y pensadas para una resolución concreta.

Desde nuestro punto de vista, de autores que escriben en castellano esto

[10] Hay algunos otros programas para ver PDF, como Ghostscript, xpdf,... pero no tienen (aún) demasiada calidad.

[11] Los argumentos *puristas* del autor para no modificar TEX hay que compensarlos con la adición de `\special`'s en el archivo fuente.

[12] En las distribuciones que lo incluyen como MikTEX o teTEX.

reabre un viejo problema. Para que LaTeX realizara correctamente el enguionado de palabras acentuadas, la solución recomendada ha sido utilizar

```
\usepackage[T1]{fontenc}
```

que cargaba las fuentes EC (European Computer Modern). Pero no existe versión de estas fuentes Type 1, por lo que un documento que utilice EC se ve muy mal en Acrobat Reader[13]. Una solución que funciona es utilizar el paquete AE (Almost European Computer Modern)[14]

```
\usepackage{ae}
```

que utiliza un conjunto de fuentes virtuales para construir una fuente con la codificación T1 con los caracteres que faltan en CM y permite el enguionado con patrones distintos del inglés. Otra solución es utilizar siempre una fuente (Times, Utopia, Palatino) que tenga todos los caracteres y sea Type 1.

Para la inclusión de gráficos se emplea la orden `\includegraphics`, estándar, cargando el paquete `graphics` con la opción `pdftex`. Permite la inclusión de gráficos en formato jpg, tiff, png, pdf y mp. Para la conversión de eps a pdf existe un *script* `epstopdf`.

El formato PDF permite incluir anotaciones en el documento, que pueden ser pequeñas notas de texto, o clips multimedia. Esta última opción depende de la plataforma, pero para los interesados se puede utilizar en pdfTeX asi:

```
\pdfannot width 5cm height 5cm depth 0cm{
/Subtype /Movie
/C [0 0 1]
/Border [0 0 3]
/Movie  <</F (sample.avi) >> }.
```

Y permitiría ver una caja de 5×5 cm con un video. Funciona con archivos `avi` `mov` y `wav` (en Windows).

El paquete hyperref

El paquete `hyperref`, escrito por Sebastian Rahtz, permite, de entrada, convertir las referencias cruzadas que normalmente utilizamos en LaTeX en hipervínculos que están activos en el documento PDF. Simplemente se carga (se recomienda que en último lugar):

```
\usepackage[pdftex]{hyperref}
```

[13] Téngase en cuenta que esto no tiene nada que ver con la codificación utilizada en el documento (`\usepackage[latin1]{inputenc}`) ni con el uso de abreviaturas definidas por `\usepackage[spanish]{babel}`.

[14] Incluido en teTeX.

Lógicamente tiene muchas más opciones, entre las que destacan los enlaces a páginas web:

\href{*URL*}{*texto*}

o las opciones que se cargan así:

```
\hypersetup{backref,pdfpagemode=FullScreen,colorlinks=true}
```

y que indican que las referencias bibliográficas incluyan un enlace para volver a la página donde fueron citadas, que el documento se abra en modo de pantalla completa y que los enlaces se vean como letras coloreadas en vez de como cajas que bordean a la letra.

Otras posibilidades de pdfTEX

El formato PDF es suficientemente rico como para ofrecer muchas más posibilidades. Sólo quiero mencionar dos: el uso de formularios y la creación de presentaciones.

En la página web de D. P. Story, sitio del proyecto AcroTEX (<http://www.math.uakron.edu/~dpstory/>), se pueden encontrar estilos de documento para crear ejercicios y pruebas de elección múltiple, además de documentos de ejemplo. También se comentan los problemas que presentan los documentos destinados a leerse en pantalla frente a los que se imprimen y se propone alguna solución.

En esa línea, el paquete `pdfscreen` (disponible en CTAN) cambia las dimensiones de una página para adecuarla a la lectura en pantalla y añade una barra de navegación. Para usarlo, en vez de cargar `hyperref` se incluye:

```
\usepackage[screen,article,sidebar]{pdfscreen}
```

Y se pueden definir algunas cosas como \emblema{*figura_de_portada*} y para la dirección del autor \emailid{*email*}.

Como uno quizá echa de menos poder crear una presentación estilo PowerPoint, en la que distintos objetos van apareciendo poco a poco en una diapositiva, se han desarrollado algunos paquetes que aprovechan las cualidades del visualizador Acrobat Reader. Se recogen a continuación las soluciones que el autor conoce en este momento, animando a los interesados a indagar en ellas. Apareció primero P^4 (PDF Presentation Post Processor) que define fondos y pausas en el documento, y tras crear el PDF este programa (escrito en Java) crea esos objetos para formar una presentación, que utilizando por ejemplo el paquete foilTEX pone a nuestro alcance toda la potencia de LATEX y el atractivo de una *presentación multimedia*. Se puede conseguir en <http://www-sp.iti.informatik.tu-darmstadt.de/software/ppower4/>.

Otras buenas soluciones para crear presentaciones dinámicas en PDF utilizando estilos existentes como `foilTeX` o `seminar` modificados son TeXpower (<http://lrb.cs.uni-dortmund.de/~lehmke/TeXPower/>), Prosper (<http://prosper.sourceforge.net>) o los estilos `pdfslide` y `pdfscreen`, disponibles en CTAN.

Estos últimos desarrollos deben completarse y mejorarse, pero muestran una vez más que TEX y sus derivados no se quedaron en aquella galaxia lejana, ni murieron hace muchos, muchos años.

Referencias

[1] M Goossens y S. Rahtz *The LateX Web Companion,* Addison Wesley Longman (1999).

TEXemplares 2/2001, págs. 32-36

Diccionario en castellano para TEX

Juan Luis Varona
Departamento de Matemáticas y Computación, Universidad de La Rioja,
C/ Luis de Ulloa s/n, 26004 Logroño, ESPAÑA.
e-mail: <jvarona@dmc.unirioja.es>
URL: <http://www.unirioja.es/dptos/dmc/jvarona/welcome.html>

Resumen

Se presenta un diccionario electrónico en castellano destinado a ser usado, fundamentalmente, por verificadores ortográficos de TEX.

El diccionario es una actualización del que estaba disponible hasta ahora y que ya se emplea, por ejemplo, en Excalibur para Macintosh, en WinEdt para Windows, y en amSpell para MS-DOS. Consiste en una recopilación de casi 300 mil palabras.

1. Verificadores ortográficos

Cuando escribimos un texto, sea del tipo que sea, es bastante común que, por mucho cuidado que pongamos, cometamos alguna errata. No sólo es posible que nos equivoquemos al recordar la ortografía de una palabra, sino que nuestra impericia escribiendo nos lleva a menudo a cambiar unas letras por otras al teclear. Para intentar minimizar esto en lo posible, unos programas llamados verificadores ortográficos vienen en nuestra ayuda.

Un verificador ortográfico comprueba que cada palabra de un texto es una de las palabras posibles en el idioma. Obviamente, un verificador nunca descubrirá un error cuando, al escribir mal una palabra, hemos escrito otra también existente, incluso aunque no tenga ningún sentido gramatical en la frase en que aparece.

En lo que a nosotros —usuarios de TEX— respecta, es importante tener en cuenta que un verificador ortográfico que quiera ser útil para comprobar documentos escritos en TEX debe comprender, al menos mínimamente, la sintaxis del lenguaje: de lo contrario, marcará como errores los comandos de TEX, las fórmulas, etc. Así pues, deberemos utilizar un verificador ortográfico especializado en TEX.

Por otra parte, el diccionario de palabras admisibles puede estar construido de dos maneras, lo que da lugar a dos tipos de verificadores ortográficos. La primera consiste simplemente en una enumeración de todas las palabras posibles. Así, cada palabra debe aparecer con todas sus variaciones permitidas al añadirle distintos prefijos y sufijos. Por ejemplo, los sustantivos y adjetivos deben aparecer en masculino, femenino, singular y plural; y los verbos conjugados.

La segunda no es una simple lista de palabras con todas sus derivaciones posibles. Aquí, podíamos pensar que en el diccionario tenemos sólo la raíz de la palabra y, además, unas reglas de derivación. Por ejemplo, para cada sustantivo deben aparecer las reglas de cómo formar masculinos, femeninos y plurales; y cada verbo debe ir asociado con sus reglas de conjugación.

Programas del primer tipo son Excalibur para Macintosh, amSpell para PC con MS-DOS y WinEdt para Windows. Del segundo tipo, podemos destacar ispell, que está disponible para equipos unix.

En general, los del primer tipo son más sencillos, pero requieren mayores ficheros de palabras. Los del segundo tipo requieren un trabajo de diseño y programación más importante, pero pueden llegar a ser más efectivos. Por ejemplo, para incluir aumentativos y diminutivos sólo hace falta añadir unas pocas reglas. En uno de los del primer tipo requiere incluir en la lista cada palabra en aumentativo y diminutivo.

Por otra parte, unos idiomas se adaptan mejor que otros a utilizar verificadores de uno u otro tipo. Por ejemplo, en inglés no hay femeninos ni plurales de adjetivos, y los verbos se conjugan muy poco. Así, no merece mucho la pena el esfuerzo de programación que se necesita para utilizar verificadores del segundo tipo. El español, en cambio, tiene gran cantidad de formas verbales que, en un verificador del primer tipo, es necesario incluir individualmente. Otros idiomas, como el alemán y el vasco, aun tienen más mecanismos de formación de palabras, como la declinación y la yuxtaposición (que puede llegar a ser bastante común), lo que hace que el número de palabras permitidas crezca desmesuradamente. Para estos idiomas, una estrategia del segundo tipo parece ser bastante más recomen-

dable. El castellano podría ser un caso intermedio.

2. La lista de palabras en castellano

En lo que a nosotros respecta, nos vamos a ocupar aquí únicamente de verificadores ortográficos del primer tipo, de los basados en una lista de palabras. Desde hace tiempo, existen diversos programas que pueden ser usados para comprobar la corrección ortográfica de textos escritos en TEX, unos gratis y otros no excesivamente caros. Pero no había ninguna lista de palabras en español suficientemente completa y de distribución gratuita. Obviamente, sí que existían implementaciones en programas comerciales pero que, en la práctica, no servían para nada si lo que queríamos era comprobar un documento TEX. Así pues, hace ya años, el que escribe se planteó elaborar una lista lo bastante amplia como para que pudiera ser usada con comodidad.

Ahora se presenta una actualización de la lista, que contiene más de 40.000 palabras nuevas. Concretamente, la lista actual consta de 292.342 palabras, mientras que la anterior tenía 247.049. Es importante hacer notar que incluye multitud de palabras relacionadas con la matemática, física e informática, que son las disciplinas donde más se utiliza TEX. Muchas de ellas no suelen estar presentes en diccionarios de carácter general.

Es de destacar que el usuario tiene acceso al diccionario fuente y puede modificarlo a su antojo para su uso personal. Aunque sería deseable que la versión que se distribuye en general permanezca unificada. Así, las futuras mejoras y añadidos de palabras se hacen siempre sobre la misma lista. El mecanismo para indicar la versión de la lista es simple: basta indicar el número de palabras que contiene.

3. Implementaciones para diversos sistemas operativos

La lista de palabras que aquí se presenta puede usarse con los siguientes verificadores ortográficos:

Excalibur

Es un programa para Macintosh, desarrollado por Robert Gottshall y Rick Zaccone. Es gratuito. Su página web es `<http://www.eg.bucknell.edu/~excalibr/excalibur.html>`, y puede conseguirse también, mediante

ftp anónimo, en <ftp://ftp.eg.bucknell.edu/pub/mac/> (en el subdirectorio Excalibur-dictionaries/ se pueden encontrar diccionarios en multitud de idiomas). En CTAN, se encuentra en el directorio <tex-archive/systems/mac/support/excalibur/>.

Una característica de Excalibur resulta muy útil a los escritores en TEX. Una palabra que contenga letras no inglesas (acentos o eñes, en el caso del español) puede escribirse con notación de 7 bits o de 8 bits. No es este lugar para comentar las ventajas y desventajas que tiene uno y otro método, puesto que cualquier usuario medianamente avanzado las conoce. Con Excalibur, basta tener la lista de palabras escrita con notación de 8 bits. Así, si una palabra de un texto está escrita con notación de 7 bits, Excalibur se encarga de traducir los 7 bits a 8 bits (en cualquier combinación entendible por TEX) antes de comparar con las palabras de la lista.

WinEdt

Es un potente y versátil editor destinado a la creación de documentos en TEX que funciona en Windows. Entre otras cosas, tiene un verificador ortográfico. Su autor es Aleksander Simonic y es un programa *shareware*. El precio de una licencia personal para uso educativo es de 40 dólares. Su página web es <http://home.istar.ca/winedt> o <http://www.winedt.com>. En CTAN, puede encontrarse en el directorio <tex-archive/systems/win32/winedt/>. Asimismo, en el subdirectorio dict/ se pueden encontrar diccionarios en bastantes idiomas.

amSpell

Es un verificador ortográfico para documentos TEX que funciona bajo MS-DOS. Su autor es A. Merckens y es gratuito. Puede descargarse, en CTAN, en el directorio <tex-archive/support/amspell/>, aunque allí sólo aparece el diccionario inglés. El diccionario en español, que es una adaptación del que aquí se presenta realizada por Agustín Martín Domingo, puede conseguirse por ftp anónimo en <ftp://tex.unirioja.es/pub/tex/dict-pc/>.

Para finalizar, únicamente añadir que todos estos programas, y las últimas versiones del diccionario español adaptado a ellos, pueden obtenerse en la Universidad de La Rioja (de donde procede el que escribe esta nota), mediante ftp anónimo, en <ftp://tex.unirioja.es/pub/tex/>. Una vez ahí, los directorios dict-mac/, dict-win/ y dict-pc/ contienen, respecti-

vamente, los diccionarios para Macintosh (Excalibur), Windows (WinEdt) y MS-DOS (amSpell) aquí descritos.

Agradecimientos: Gracias a todos los que han colaborado notificándome erratas de la versión anterior de la lista de palabras, o enviándome nuevas palabras para incluir. En particular, a Héctor M. Monacci y a Josu Zabaleta.

TEXemplares 2/2001, págs. 58-68

Experiencia con utilidades de programación ilustrada en TEX, LATEX

E. Meléndez, R. Herrero

1. Introducción

Definición

De acuerdo a la 'literate programming FAQ', es decir, la lista de preguntas frecuentes del grupo de noticias `comp.programming.literate` dedicado a la programación ilustrada[19] [1], la programación ilustrada es la combinación de la documentación y del código fuente en una forma adecuada para la lectura por humanos. En general, esto se traduce en «explicar» de la forma más detallada posible qué acciones se están tomando en el curso de la ejecución de un programa, con qué motivo, qué algoritmos se están usando, por qué funcionan, etc., y hacerlo de tal manera que se facilite la tarea del programador propiciando que la documentación se escriba a la vez que se escriba el programa, y la de aquellas personas que deben revisar o quieren entender ese programa. Esto se consigue aunando la documentación y el código fuente del programa en el mismo fichero.

En la práctica, un código fuente escrito mediante estas técnicas está compuesto por un conjunto de trozos (*chunks*). Cada trozo consta de una sección explicativa, que puede incluir ecuaciones, figuras, tablas, referencias externas a otros volúmenes de documentación, etc., y una sección de código fuente propiamente dicho, formada por código puro y referencias a otros trozos. Un programa se encargará de generar el fichero de entrada

[19] Esta acertada traducción de *literate programming* se debe a Gabriel Valiente

para un determinado procesador de textos (TEX - LATEX en nuestro caso) dando una apariencia estética al conjunto. Otro programa eliminará las secciones explicativas y sustituirá las referencias a otros trozos por su contenido correspondiente, a fin de construir el código fuente compilable. Este proceso de sustitución de los trozos es similar a una expansión de macros, por lo que la estructuración del programa puede conseguirse de esta forma sin necesidad de usar subrutinas. El orden en que se escriben los trozos queda al arbitrio del desarrollador, que puede expandir el código en horizontal o en vertical según sus preferencias sin que la operatividad del programa se vea afectada. El formateador de textos que genera el código documentado puede establecer enlaces de hipertexto entre los trozos, por lo que resulta muy cómodo navegar por la documentación. Esta característica, unida a la capacidad de algunos visualizadores para buscar palabras clave, hace que un programa desarrollado con técnicas de programación ilustrada sea muy fácil de entender, mantener y modificar incluso para terceras personas. El apéndice B muestra un ejemplo de programa documentado mediante programación ilustrada.

El mundo de la programación ilustrada se ha organizado en un anillo de recursos en Internet llamado *Literate Programming Webring*, que puede alcanzarse desde la página matriz de una de las herramientas que se van a explicar en esta ponencia, `http://www.cs.virginia.edu/~nr/noweb`.

Origen histórico

Fue Donald E. Knuth quien creó el estilo WEB[4, 5], con la idea de que el programador pudiera proporcionar la mejor documentación posible sobre su programa, usando TEX para ello, sin tener que abandonar el entorno de programación. La combinación de los dos aspectos de un programa (su código fuente y la explicación de éste) adecuadamente combinados en el mismo documento da lugar a un sistema de programación que es mucho más útil que el lenguaje de programación y TEX por separado.

El propio programa TEX está programado, de hecho, haciendo uso de las técnicas de programación ilustrada, en concreto mediante WEB. TEX estaba, originalmente, programado en PASCAL, siendo ahora común la versión en la que este programa en PASCAL se convierte al lenguaje C.

Herramientas disponibles

La programación ilustrada, aunque en origen diseñada por Knuth, ha evolucionado para convertirse en una forma de realizar programas, no aso-

ciada necesariamente a la herramienta que él diseño. Así, existen, además del sistema WEB y sus derivados (de las que hablaremos más extensamente en el resto de esta ponencia), otro tipo de herramientas ligadas a otros sistemas de procesamiento o edición de textos. Brevemente citaremos, Literate Programming Workshop (LPW), herramienta diseñada para su uso en ordenadores Apple; CLiP, que puede usar LaTeX, TeX o cualquier procesador de textos, usando un formato especial para distinguir el código fuente del resto de la documentación; y WinWordWEB, un conjunto de macros para WinWord que proporciona un entorno básico de programación ilustrada.

Además de estas herramientas, hay disponibles programas que proporcionan un primer paso hacia la programación ilustrada convirtiendo el código fuente en un documento que puede después tratarse con TeX, LaTeX u otro tipo de procesador de textos, y que dan un formato especial a los comentarios incluidos en el fuente.

2. Herramientas concretas en TeX, LaTeX

Un gran número de herramientas usan TeX o LaTeX para la edición de la documentación. El programa padre de todos los sistemas de programación ilustrada, como se ha dicho ya, es WEB, debido a Donald E. Knuth, y originalmente escrito para PASCAL. Se pueden clasificar estas herramientas en dos grupos, en función de si dependen o no del lenguaje de programación que se vaya a usar. Dentro del grupo de sistemas 'dependiente del lenguaje de programación' se encuentran el propio WEB y sistemas como CWEB (para el lenguaje C o C++), CWEBx3.0 (ANSI C), mCWEB (C, C++), IMPACT (C, C++), FWEB (FORTRAN), MWEB (Modula-2) y SchemeWEB (LISP/Scheme). Siguiendo las técnicas de programación ilustrada, se ha diseñado también un estilo de programación de JAVA que permite que sea el propio compilador el que extraiga la documentación del código fuente. [9] Como se ve, predominan las herramientas para C y C++. Al haberse concebido estas herramientas para un lenguaje de programación concreto, en el texto resultante las porciones que presentan el código fuente tienen un tratamiento especial, incluyendo, por ejemplo, un tipo de fuente específico, símbolos para los operadores y las constantes más usadas, etc. CWEB ha sido desarrollado por D. E. Knuth y Silvio Levy, y hablaremos de él más detalladamente.

En el otro grupo las herramientas no están asociadas a un lenguaje de programación particular. Éstas, en general, no dan formato a los trozos de

fuente que aparecen en el texto de la documentación. Las herramientas de este grupo incluyen FunnelWeb, noweb, nuweb, ProTeX, SpideryWEB.

`WEB` **y** `cweb`

El sistema *WEB* data de 1981, y fue originalmente diseñado para implantar la programación ilustrada usando PASCAL. Posteriormente, Silvio Levy, en colaboración con el propio Knuth, creó una versión para C denominada cweb y que tiene las mismas capacidades y características que el programa original. El sistema *WEB* está diseñado para trabajar con TEX simple, aunque existen extensiones para LATEX. Existe una clase especial para ficheros de tipo *CWEB* creada por Joachim Schrod, dedicada a la elaboración de documentos *CWEB* en los que se puedan usar todas las utilidades de LATEX.

El sistema diseñado por Knuth consta de dos programas separados, a saber, ctangle que extrae el código fuente[20] , y cweave que extrae la documentación[21].

Al aplicar *ctangle* sobre un documento se extrae únicamente el código fuente, entendiendo por tal el fichero principal, que tendrá el mismo nombre que el que se está procesando y extensión `.c` salvo que se especifique otro al ejecutar el programa, y también los ficheros auxiliares que se hayan indicado en el propio fuente de *WEB*. Para ayudar en la depuración de los programas generados, se incluyen directivas que encaminan al programa de depuración (*debugger*) al fichero *WEB* en lugar de al fichero compilado. Este comportamiento puede suprimirse mediante la opción `-l` de *ctangle*.

cweave convierte el fichero `WEB` en un fichero que se puede procesar con TEX o LATEX. De forma automática, cweave incluye una relación de las variables usadas en el código, indicando dónde se usa cada una y cuándo se define, y el índice del documento. Esto puede evitarse usando la opción `-x`.

Un documento *WEB* está estructurado en secciones, también llamados bloques (*chunks*) que contienen tres partes:

- la de documentación, que respeta la sintaxis y estructuras de TEX o LATEX y proporciona la explicación del código de esa sección, que comienza por `@\` o `@*` (que indica el comienzo de una sección);

[20] *to tangle* puede traducirse por 'enrollar, liar'
[21] *to weave* tiene el significado de 'tejer'

- una parte intermedia que contiene definiciones y macros del lenguaje C, usada para simplificar la estructura del código, que si existe contiene líneas que comienzan por `@d`, y que se situará al principio del fichero que contiene el código fuente; y
- el código en C, que contiene la parte del programa que se combinará con las demás para formar el fichero que contiene el programa completo, que comienza por la estructura '`@<`' seguido de texto seguido de '`@>=`'.

Al comienzo de cada bloque marcado con `@*` se puede indicar el nivel de seccionamiento del bloque por un número decimal después del asterisco. El título de la sección se incluye a continuación, y debe terminar en un punto (`.`). El nombre de la zona de código en C del bloque (lo que queda entre los `@<...@>=`) puede abreviarse poniendo sólo una cadena inicial, que deberá ser suficiente para identificar unívocamente el bloque, y tres puntos (suspensivos). Para situar un bloque de programa dentro de otro se hace referencia a él por su nombre encerrado en `@<...@>`, es decir, omitiendo el `=`.

`cweave` identifica los comentarios incluidos en el código fuente, y los procesa adecuadamente (con TEX). Es la forma de incluir texto en la parte de código de la documentación. No es necesario ningún código especial para singularizar estas porciones de TEX en el código fuente, dado que están ya en un entorno especial (el comentario).

De forma recíproca, cuando la descripción de un bloque hace referencia a una variable existente en el programa, `cweave` le da el tratamiento adecuado de dos formas: una, imprimiéndola en el mismo tipo que el código fuente; otra, incluyendo en la tabla de referencias el número del bloque en el que se ha citado la variable; además, subraya la cita que corresponde a la definición o declaración de cada identificador. Para identificar cuándo en una zona de texto se está hablando de una variable, se debe encerrar ésta entre barras verticales (`|...|`, de forma similar a la redefinición de `\verb` en muchos documentos de LATEX). Los nombres de la zona de código C de cada sección se consideran parte del código, y por tanto deben encerrarse entre barras cuando se haga referencia a ellas en la zona de texto.

`cweave` sólo impone una restricción en la escritura de la parte de código: que el símbolo `@` debe escribirse `@@`. Esto es así porque las macros que entiende `cweb` comienzan por `@`.

`cweb` admite un número de códigos que permiten mantener un control preciso sobre el formato del documento. `@f` y `@s` son códigos que indican

a TEXcómo ha de imprimir los identificadores. En concreto, `@f l r` indica que el identificador `l` debe tratarse en la misma forma que se trata `r`. cweave entiende las sentencias de definición (**typedef**), por lo que estos se usan relativamente poco.

Ciertos códigos permiten dirigir la estructuración del programa en C, señalando el comienzo del programa, y permitiendo que una determinada sección se encamine a un fichero distinto (por ejemplo, para construir ficheros de cabecero (`.h`).

Otro grupo de códigos gestiona cómo se transfiere la información a las partes de documentación y de fuente. Existe la posibilidad de introducir palabras clave en el índice, e incluso comentarios en el fuente de *web* que no se transferirán a la salida.

WEB modifica el fichero fuente que aparece en la parte C, por ejemplo para introducir de forma conveniente los códigos ASCII en el rango 128-255.

Finalmente, cweb admite códigos para modificar la apariencia del código fuente que aparecerá en la documentación, por ejemplo, para establecer los cambios de linea adecuados.

cweave y ctangle están diseñados para trabajar con dos ficheros, llamados el fichero web y el el fichero de «cambios» (fichero ch. El fichero de cambios contiene datos que modifican el fichero web. El fichero de cambios contiene cero o más cambios, que consisten en bloques de la forma '`@x`líneas antiguas`@y`nuevas líneas`@z`'. Los códigos `@x`, `@y` y `@z` sólo pueden aparecer al comienzo de una línea y en el fichero de cambios.

El código `@i` seguido del nombre de un fichero al comienzo de la línea permite insertar ese fichero *WEB* en el punto donde aquel código aparece.

Para que TEX entienda los formatos introducidos por cweave se debe incluir el fichero `cwebmac.tex`, que proporciona las macros adecuadas.

noweb

El sistema *WEB* presenta un alto grado de sofisticación y un gran número de secuencias de control para dar formato a la documentación de un programa escrito en PASCAL (*WEB*) o en C (*CWEB*). La adaptación a otros lenguajes de programación no es una tarea simple, ya que *CWEB* reconoce parte de la sintaxis del lenguaje que está tratando. El grado de complejidad en cuanto a los códigos de formato es alto, como puede verse en la sección precedente. Con el tiempo esta situación ha evolucionado en dos caminos. Uno es el diseño de herramientas que extienden *WEB* tanto para cubrir otros lenguajes como para facilitar su uso. El otro es la abstracción

del lenguaje de programación para hacer herramientas 'universales' que puedan usarse con cualquier lenguaje de programación. Éste es el caso de noweb. noweb parte de la premisa (usual en el mundo UNIX) de que las no deben llegar a un grado de sofisticación demasiado elevado, de forma que es mejor escribir dos herramientas cuya unión proporcione el resultado requerido que una sola herramienta complicada de usar. noweb se apoya también en la concatenación de programas en UNIX, por la cual la salida de un programa puede alimentar directamente a otro. Es lo que en inglés se llama *pipelining* (encadenado de órdenes).

El conjunto de utilidades *noweb* se basa en cuatro herramientas principales,

noweave convierte el fichero *noweb* en un fichero adecuado para su procesamiento con TEX o LATEX. En el primer caso, un conjunto especial de macros (nwmac) proporciona las definiciones adecuadas para el seccionamiento. Las opciones `-tex` y `-latex` condicionan este comportamiento. Otros formateadores de la salida de noweave se consiguen con las opciones `-html`, `latex+htl` (para su uso posterior con latex2html), -troff.

notangle extrae el código fuente, copiando literalmente las partes de código a la salida estándar del programa.

noweb que es una versión simplificada de noweave

nountangle que convierte un fichero *noweb* en código fuente en el que las descripciones que aparecen en las zonas de texto de cada bloque se han convertido en comentarios en el lenguaje de programación correspondiente.

La sintaxis de noweb es similar a la de *WEB*. En cuanto a la forma de escribir el fichero *NOWEB*, las diferencias son como sigue. Cada bloque comienza por el código `@\␣`, pero no existe la orden `@*`, ya que el seccionamiento viene dado directamente por el seccionamiento normal en LATEX. No se distingue la parte intermedia de definiciones y macros. La parte de código comienza con la marca '<< nombre del bloque >>=' . Las referencias a cada bloque se hacen omitiendo el =. Las referencias a variables del programa en el texto de la documentación deben ir entre las marcas `[[...]]`.

En su comportamiento, noweb es netamente distinto de *WEB*. Se ha abstraído el marcado y la interacción con cada lenguaje de programación,

construyéndose filtros que proporcionan las definiciones y marcas adecuadas para obtener el índice y las referencias de uso de los bloques. El propio autor de noweb ha programado filtros para los lenguajes de programación más comunes y que más le han convenido (C, PASCAL, ICON, YACC, PROMELA, ML estándar, LRTL y TEX), de forma que no hay merma en las capacidades en cuanto a la calidad de la información proporcionada por el programa. Un filtro adecuado proporciona, análogamente a como se hace en *WEB*, el índice de definiciones de variables usadas en el código y el cruce de referencias de bloques, indicando dentro de cada uno dónde se usan los identificadores que define cada bloque, y qué identificadores se definen. Cuando no existe un filtro adecuado para el lenguaje de programación que se está usando, una línea de que tenga la forma `@ %defs` *identificadores* indica que en el bloque precedente se definen los identificadores que en ella se expresan, haciendo posible el marcado. Fruto de esta abstracción del lenguaje de programación es el hecho de que puedan coexistir los ficheros fuente de varios lenguajes dentro del mismo fichero de *noweb*. Esto simplifica notablemente la generación de ficheros fuente para programas que necesitan la coordinación de varios lenguajes. El caso paradigmático es la interpretación de ficheros por medio de un analizador léxico (por ejemplo, flex) y un analizador sintáctico (tal como bison). En su uso tradicional, estos dos programas necesitan ficheros distintos, pero por su naturaleza fuertemente interdependientes. La posibilidad de construirlos simultáneamente simplifica grandemente la programación. Esta filosofía necesita que se puedan usar distintos bloques como raíz de cada programa, raíz a partir de la que se construirá el fichero fuente de aquél. Es por tanto legal en el procesamiento de un fichero *noweb* que no se usen algunos bloques. La referencia a un bloque inexistente se considera un error.

Por sí solo *noweb* no proporciona una presentación sofisticada del código fuente, como sí lo hace *WEB*. Las partes de código se imprimen en tipo `máquina de escribir`, y no se incluyen símbolos especiales para, los operadores de comparación, constantes numéricas, etc. Sin embargo, la naturaleza modular que es la base de la filosofía de uso de *noweb* permite que se puedan hacer filtros para conseguir este formato preciosista del código fuente.

Como parte de su filosofía modular, puede indicarse a noweave (con la opción `-macro`) que su salida se construya especialmente para usar un filtro posterior. Con esta opción, se incluyen marcas especiales ('palabras clave estructurales', en la definición de *noweb*), que singularizan cada una de las partes del fichero *noweb* y permiten que el filtro posterior reconozca

cada parte y le dé el tratamiento que le corresponda.

3. Conclusiones y sugerencias

La programación ilustrada es una técnica que permite documentar exhaustivamente el código fuente de un programa. En el entorno TEX - LATEX se dispone de varias herramientas para programación ilustrada, de las cuales las más significativas son WEB, cweb y noweb.

Las herramientas de programación ilustrada disponibles en TEX - LATEX están orientadas al formato de página. Sin embargo, creemos que la orientación a pantalla, sin discontinuidades, es más apropiada. En este formato, resultaría interesante poder desplegar y plegar las referencias a los trozos y reemplazarlas por sus contenidos, de manera análoga a los directorios en un gestor de archivos gráfico. Los autores tienen la intención de dotar a *noweb* de esta capacidad.

Otra herramienta de gran interés sería un depurador en el que el código fuente apareciera con su correspondiente documentación, hipertexto entre trozos que se podrían plegar y desplegar, etc. Parece éste un proyecto complicado, que los autores no tienen intención de acometer.

Referencias

[1] D. B. Thompson. The Literate Programming FAQ, `CTAN:help/Lit-Prog-FAQ`

[2] P. J. Denning Announcing literate programming. *Communications to the ACM*, 30(7):593, July 1987.

[3] K. Guntermann and J. Schrod. `WEB` adapted to C. *TUGBoat*, 7(3):134-137, Oct. 1986.

[4] D. E. Knuth. The `WEB` system of structured documentation. Technical Report 980, Stanford Computer Science, Stanford, Califoria, Sept. 1983.

[5] D. E. Knuth. *Literate Programming*, Stanford University, Stanford, California, 1992.

[6] S. Levy. `WEB` adapted to C, another approach. *TUGBoat*, 8(1):12-13, Abril, 1987.

[7] N. Ramsey. Literate programming: Weaving a language-independent WEB. *Comminucations to the ACM*, 32(9):1051-1055, Sept. 1989

[8] N. Ramsey. Literate programming simplficed. *IEEE Software* 11(5), 97.105, Sept. 1994.

[9] Página raíz de HotJava, `http://java.sun.com/`

A. Recursos en Internet

Programación ilustrada

Además del anillo WEB de programación ilustrada y del grupo de noticias ya mencionado, los recursos de programación ilustrada pueden encontrarse en `ftp://ftp.th-darmstadt.de/programming/literate-programming` (LPA).

Herramientas de programación ilustrada

La lista siguiente se ha confeccionado a partir de la 'FAQ' de programación ilustrada. Contiene las direcciones donde se pueden encontrar los programas que se mencionan en la ponencia.

CWEBx3.0 `ftp://ftp.cwi.nl/pub/cweb`

FWEB `LPA:fweb`, `CTAN:/web/fweb`

SchemeWEB `LPA:lisp`, `CTAN:web/schemeweb`

SpideryWEB `LPA:spiderweb`

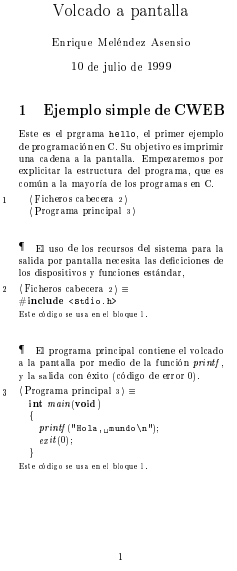
noweb `http://www.cs.virginia.edu/~nr/noweb`

WEB `LPA:pascal`

CWEB `LPA:c,c++`, `CTAN:web/c_cpp/cweb`

B. Ejemplo

```
% Ejemplo simplicísimo de
% programación CWEB.
\documentclass[language=spanish]{cweb}
\usepackage[T1]{fontenc}
\usepackage{spanish}
\begin{document}
\title{Volcado a pantalla}
\author{Enrique Meléndez Asensio}
\date{10 de julio de 1999}
\maketitle

@* Ejemplo simple de CWEB.
Este es el programa \texttt{hello},
el primer ejemplo de programación en C.
Su objetivo es imprimir una cadena a la
pantalla. Empezaremos por explicitar la
estructura del programa, que es común a
la mayoría de los programas en  C.

@c
@<Ficheros cabecera@>@/
@<Programa principal@>

@ El uso de los recursos del sistema
para la salida por pantalla necesita
las definiciones de los dispositivos
y funciones estándar,

@<Ficheros...@>=
#include <stdio.h>

@ El programa principal contiene el
volcado a la pantalla por medio de
la función |printf| y la salida con
éxito (código de error 0).

@<Programa...@>=
int
main(void)
{
printf("Hola, mundo\n");
exit(0);
}
@ \end{document}
```



Volcado a pantalla

Enrique Meléndez Asensio

10 de julio de 1999

1 Ejemplo simple de CWEB

Este es el prgrama `hello`, el primer ejemplo de programación en C. Su objetivo es imprimir una cadena a la pantalla. Empezaremos por explicitar la estructura del programa, que es común a la mayoría de los programas en C.

1 ⟨ Ficheros cabecera 2 ⟩
⟨ Programa principal 3 ⟩

¶ El uso de los recursos del sistema para la salida por pantalla necesita las deficiciones de los dispositivos y funciones estándar,

2 ⟨ Ficheros cabecera 2 ⟩ ≡
#**include** <stdio.h>
Este código se usa en el bloque 1.

¶ El programa principal contiene el volcado a la pantalla por medio de la función *printf*, y la salida con éxito (código de error 0).

3 ⟨ Programa principal 3 ⟩ ≡
int *main*(**void**)
{
printf("Hola, mundo\n");
exit(0);
}
Este código se usa en el bloque 1.

1

TEXemplares

Año 3.º, número 2, primer trimestre 2000

Índice