TEX: Edición Programable de Matemáticas

Joseph C. Várilly

Universidad de Costa Rica, 2060 San José, Costa Rica

TEX es un software para levantamiento y fotocomposición de textos técnicos de alta calidad gráfica. Actualmente hay diversas versiones de TEX para computadoras grandes y pequeñas. Algunas de las versiones mas usadas en la actualidad son PC TEX y emTEX para sistemas PC-DOS, MikTEX y DviWindo para sistemas basadas en Windows, *Textures* y OzTEX para Macintosh, teTEX para sistemas Unix.

El manual principal es *The TEXbook* de Donald E. Knuth, el creador del sistema. Ese manual está organizado en tres niveles paralelos: (1) para usuarios iniciales, (2) para usuarios intermedios, y (3) para expertos. Este documento no pretende reemplazar el *TEXbook*, sino proporcionar una introducción en español en sus niveles básico e intermedio para personas que pretenden usar TEX para levantar escritos con una alta proporción de simbología matemática. Los detalles que aquí se omiten pueden encontrarse en alguna parte del *TEXbook*, que es el libro de referencia canónica.

TEX es una marca registrada del American Mathematical Society. El nombre puede pronunciarse *tej* o *tecs* según el gusto de cada quien.

1 Los Caracteres de TEX

1.1 Caracteres del teclado

El teclado de una computadora tiene al menos 94 caracteres visibles, mayúsculas y minúsculas. Podemos clasificar estas teclas así:

Letras:	`A, B,..., Z; a, b,..., z`
Dígitos:	`0, 1, 2,..., 9`
Signos de puntuación:	`. , ; : ! ? " ‘ ’ ( ) [ ]`
Signos aritméticos:	`+ - * = / | < > @`
Caracteres especiales:	`\ { } $ & # ^ _ ~ %`

Figura 1.1. Caracteres visibles del teclado inglés

Los diez caracteres especiales *cumplen funciones de control* en TEX, y por ende deben usarse con cuidado. Los otros caracteres (letras, dígitos, signos de puntuación y aritméticos) se reproducen directamente en impresión. Los idiomas europeos incluyen otros caracteres que llevan acentos ortográficos (como las letras ñ y Ñ en español; vocales acentuados como å, é, î, ò, ü; y letras especiales como æ, ç, ø, ß). La mayoría de estos se obtienen en TEX con procedimientos sencillos de acentuación. Los caracteres escritos \$, &, #, % pueden obtenerse mediante *códigos*, al teclear `\$`, `\&`, `\#` y `\%` respectivamente.

Los diez caracteres especiales se usan para *controlar* el levantamiento de texto. Brevemente, sus funciones son:

`\`	es el carácter inicial de un *comando*;
`{ }`	abren y cierran *grupos* o bloques de código;
`$`	abre y cierra una *expresión matemática*;
`&`	funciona como un *tabulador* en texto alineado;
`#`	señala un *parámetro* de un macro;
`^ _`	se usan para indicar *exponentes* y *subíndices*;
`~`	es una *liga* que controla cortes de renglón;
`%`	es el código para señalar un *comentario* en el archivo fuente.

Figura 1.2. Funciones de los diez caracteres especiales

Retomaremos cada una de estas funciones especiales oportunamente. Por ahora, basta señalar que los caracteres de esta lista están reservados para usos especiales y que al principio hay que tener especial cuidado al teclear texto que contenga cualquiera de ellos.

Hay algunos otros caracteres que también se obtienen directamente del teclado, que son fáciles de olvidar, porque son *invisibles*: son el espacio en blanco producido al tocar la barra de espaciamiento, el retorno de carro, la tabulación y el corte de página. (El corte de página se obtiene de ciertos procesadores de palabras; su efecto en TEX es equivalente a dos retornos de carro.) Para denotar un toque explícito a estas teclas, usaremos en estos apuntes los símbolos siguientes:

Símbolo	*Significado*
␣	un espacio tecleado (= un toque a la barra),
↵	un retorno de carro,
↹	una tabulación.

Figura 1.3. Caracteres invisibles del teclado

1.2 Caracteres de texto impreso

La primeras versiones de TEX aceptaban solamente los caracteres del abecedario inglés directamente, obligando al usuario a teclear combinaciones como `\'a` para obtener caracteres europeos como á; pero a partir de 1990, se dispone de una versión de TEX que permite teclear `á` directamente. Este guía supone el uso de una implementación de TEX (como *Textures* 2.0 u otro similar) que incluye los archivos necesarios para facilitar el ingreso de cualquier caracter del teclado del usuario.

Una hoja de texto corriente hace uso de todos los caracteres antes mencionados, además de ciertos otros caracteres producidos en TEX como *caracteres compuestos*: son las comillas, los guiones, y las ligaduras.

1.2.1. Comillas.

Un teclado normal tiene los caracteres ‘ ’ “ y ”. Su disponibilidad depende del sistema operativo, y por lo tanto TEX ofrece un modo alternativo de generar las comillas que los hace transferibles entre diversos sistemas. TEX normalmente usa un *apóstrofe* ’ que produce ’ en impresión, y la tecla ‘ produce ‘ . La tecla " produce ” pero para obtener “ hay que teclear ‘ dos veces; simétricamente, si se teclea ’ dos veces, también se obtiene ”. En resumen:

Se teclea	*Se imprime*
‘	una comilla simple inicial: ‘
’	una comilla simple final: ’
‘‘	comillas dobles iniciales: “
’’ o bien "	comillas dobles finales: ”

Figura 1.4. Tecleo de comillas en TEX

1.2.2. Guiones.

En un teclado hay solo un guión, pero en impresión hay cuatro: el guión ordinario (-), una barra corta (–), una barra larga (—), y el signo menos (−). Se usan barras cortas entre dígitos numéricos (como en: “las páginas 46–49”) y barras largas entre palabras — con la función de un paréntesis. Para distinguir las alternativas, se teclean uno, dos o tres guiones sucesivos; para un signo menos, se teclea un guión entre signos de dólar (esto es un *efecto matemático*, que veremos más adelante). En resumen:

Se teclea	*Se imprime*
`-`	un guión ordinario: -
`--`	una barra corta: –
`---`	una barra larga: —
`$-$`	un signo menos: −

Figura 1.5. Tecleo de guiones en TEX

1.2.3. Ligaduras.

Algunas pares o ternas de letras tecleadas se combinan en impresión para producir un carácter compuesto. Ellos son `ff`, `fi`, `fl`, `ffi` y `ffl`, que se imprimen como ff, fi, fl, ffi, y ffl.

El programa TEX posee (como parte de su estructura interna) una unidad lectora (un "scanner", para usar un anglicismo de moda) que lea los caracteres tecleados y *automáticamente* combina los pares o ternas que forman estas *ligaduras*. El tecladista no tiene que hacer nada especial.

Los caracteres que no son letras también pueden formar ligaduras; de hecho, las barras se obtienen al aplicar este artificio al par `--` y a la terna `---` ; también, las comillas dobles se arman así de `‘‘` y `’’`.

Para imprimir texto en español, se usan otras dos ligaduras: `?‘` produce el signo de interrogación inicial ¿ mientras `!‘` produce el signo de admiración inicial ¡.

1.3 Caracteres compuestos especiales

Ciertas variantes del abecedario romano han surgido en diversos idiomas. El latín tiene las letras æ y œ, que son contracciones de ae y oe. El danés tiene æ y además å y ø. El polaco tiene la letra Ł. El alemán posee ß, una contracción de ſz en la antigua tipografía "fraktur" (no debe confundirse con β, la "beta" griega). Todas estas letras se obtienen al teclear comandos: el símbolo \ seguido por una o dos letras apropiadas. (Alternativamente, se puede preparar *Textures* para que acepte las teclas correspondientes directamente, con la lectura del archivo `option_keys`.) La lista completa es:

Å, å	⟵	`\AA`, `\aa`	Ø, ø	⟵	`\O`, `\o`
Æ, æ	⟵	`\AE`, `\ae`	Ł, ł	⟵	`\L`, `\l`
Œ, œ	⟵	`\OE`, `\oe`	ß	⟵	`\ss`

Figura 1.6. Caracteres compuestos

1.4 Acentos ortográficos

TEX dispone de una gran variedad de *acentos* para textos en español, portugués, francés, italiano, alemán, etcétera. Hay un total de 14 acentos ortográficos en Plain TEX, que se obtienen mediante comandos:

`\’o`	ó	(acento agudo)	`\u e`	ĕ	(acento breve)
`\‘a`	à	(acento grave)	`\v u`	ǔ	(háček)
`\^e`	ê	(circunflejo)	`\H o`	ő	(diéresis húngaro)
`\"u`	ü	(diéresis)	`\c c`	ç	(cedilla)
`\~n`	ñ	(onda)	`\d o`	ọ	(punto debajo)
`\=o`	ō	(barra)	`\b o`	o̲	(barra debajo)
`\.a`	ȧ	(punto)	`\t oo`	o͡o	(atadura)

Figura 1.7. Los acentos de texto en TEX

Los primeros cuatro acentos pueden colocarse sobre vocales, directamente del teclado. Comúnmente, se dispone de archivos auxiliares que permiten incorporar estos caracteres (`ó`, `à`, `ê`, `ü`, etc.) directamente en el tecleo de un archivo para TEX; en español y catalán, se pueden usar `ñ`, `Ñ`, `ç`, `Ç` directamente del teclado. (Con las antiguas versiones de TEX, la versión larga de la Figura 1.7 da el mismo efecto final, al precio de un trabajo más extenso y tedioso por parte del tecladista.) Para casos excepcionales no previstos en un teclado hispanográfico, los acentos de la Figura 1.7 siempre quedan disponibles: tal es el caso, por ejemplo, de las letras rumanas ş (`ş`) y ţ (`ţ`) y las portuguesas ã (`ã`) y õ (`õ`).

2 Comandos en TEX

2.1 Símbolos de control

Los comandos de TEX empiezan con el carácter \ (el "carácter de escape"),* que llamaremos **vara**, para distinguirlo de las otras rayas recta | y oblicua /. Una vara seguida inmediatamente por cualquier carácter *que no sea una letra* (minúscula o mayúscula) *del abecedario inglés* se llama un **símbolo de control**.

Los símbolos de control tienen varios efectos; he aquí el catálogo:

Caracteres especiales:	`\# \$ \% \& \_`
Acentos:	`\' \` \^ \" \~ \= \.`
Efectos matemáticos:	`\{ \} \* \> \, \; \!`
Efectos especiales:	`\␣ \+ \- \/`
Comandos no definidos:	`\0 \1 ... \9, \: \? \< \( \) \[ \] \\`

Figura 2.1. Símbolos de control

La última lista consta de *posibles* símbolos de control que no tienen función asignada en Plain TEX. Veremos luego que estos símbolos pueden usarse como *macros* o sinónimos cortos, para procedimientos de relativa complejidad. Los caracteres de un teclado extendido también dan lugar a otros posibles símbolos de control, como `\á`, `\é`, `\ñ`, etc., que no tienen definición asignada en Plain TEX.

Algunos estilos de formato, más sofisticados que Plain TEX, asignan funciones a los símbolos de control que Plain TEX deja indefinidos. Por ejemplo, $\mathcal{AMS}$-TEX define `\:` y `\\`, a la vez que remueve las definiciones de `\=` y `\>`, dejando libre estos símbolos de control para otros usos. LATEX establece definiciones para `\(`, `\)`, `\[`, `\]` y `\\`. Veremos eventualmente cómo se pueden hacer estos ajustes a las funciones de cada símbolo de control.

El comando \␣ (vara, toque a la barra) es un espacio en blanco *explícito* que produce un espacio en blanco en impresión. Para adoptar un convenio universal aplicable a los diversos procesadores de texto existentes, Plain TEX considera \↵ y \↹ como sinónimos de \␣. Así, una vara seguido por un carácter invisible siempre produce un espacio en impresión; en particular, una vara al final de un renglón del archivo fuente genera un \↵ y produce un espacio en blanco impreso.

2.2 Palabras de control

El segundo tipo de comando en TEX es la *palabra de control*. Una palabra de control es una vara \ seguido inmediatamente por una *palabra*, es decir, por una cadena ininterrumpida de letras. Una *letra*, para los efectos de esta discusión, es un carácter, minúscula o mayúscula, del abecedario inglés: `A, B, C,..., Z, a, b, c,..., z`. La "palabra" puede constar de una sola letra, dos letras, tres letras, ... hasta $(n-1)$ letras, si n es el máximo número de caracteres que cabe en una línea del archivo fuente.†

TEX, un software moderno, distingue entre letras mayúsculas y minúsculas. Luego las palabras de control `\psi`, `\Psi`, `\pSi`, `\psI`, `\PSi`, `\PsI`, `\pSI`, `\PSI` representan ocho comandos distintos. Ahora `\psi` y `\Psi` están definidos por Plain TEX (son las letras griegas ψ y Ψ) pero los otros seis carecen de definición y pueden usarse como macros sin incomodar a Plain TEX. El logo TEX se produce con el comando `\TeX`, cuya extraña combinación de mayúsculas y minúsculas sirve como marca de distinción.

Una *palabra* es cualquier cadena ininterrumpida de letras; luego, cualquier cosa que no sea una letra, como un espacio en blanco ␣, un dígito numérico, un signo de puntuación, un ↵, un ↹, u otro \, termina la "palabra". Luego `\fea`, `\espacio`, `\yonosequeestapasando`, `\Que`, `\QUE`, `\quE`,

* No se trata de la "tecla de escape" `ESC` que poseen algunos teclados.

† Para poder transmitir archivos por correo electrónico, sin riesgo de corrupción, se aconseja limitarse a 80 caracteres por renglón.

`\A`, `\a`, podrían ser "palabras de control"; pero `\quie bra`, `\Parte2`, `\uno-a-uno`, `\er,ror` no son palabras de control.

Para asegurar que los espacios en blanco sirvan para terminar palabras de control, TEX tiene una *regla* muy importante: **Cuando un espacio en blanco sigue una palabra de control, TEX lo desecha**; y sigue desechando espacios en blanco hasta llegar al próximo carácter visible. A modo de ejemplo, la palabra de control `\TeX` produce el logo 'TEX', pero

al teclear: `\TeX es muy feo`, se obtiene: TEXes muy feo

ya que se bota el espacio después de `\TeX`. Esto ocurre pocas veces, pero hay un remedio sencillo: el *símbolo* de control \␣ produce un espacio en impresión *que no desaparece*. Luego se debe teclear: `\TeX\ es muy feo`.

Ahora, solo hay que *aprender los significados* de todos los comandos que Plain TEX entiende, y dispondremos de una maquinaria capaz de producir los más variados efectos en impresión!

2.3 Efectos de los comandos

Pasamos de la sintaxis a la semántica de los comandos. Podemos clasificar los comandos según sus efectos, así:

- ◇ Comandos que efectúan cambios ambientales;
- ◇ Comandos que actúan sobre "argumentos" que los siguen;
- ◇ Comandos que poseen un efecto autónomo.

2.3.1. Cambios ambientales.

Ejemplos de estos son los comandos que cambian el tipo de letra en uso (de romano a itálico, o viceversa), o los que modifican los márgenes del texto (para una cita textual, quizás), o los que preparan un sitio para la inserción de una figura o tabla. Hay pocos comandos de este tipo. El efecto surtido continúa en vigor hasta que sea cancelado explícitamente.

2.3.2. Comandos que toman argumentos.

Otros comandos actúan sobre el carácter (o el bloque de texto) que sigue: los acentos funcionan de este modo. Estos comandos requieren uno o más "argumentos" que deben seguirlos inmediatamente. Por ejemplo, la ñ se produce con `\~n`: aquí la letra `n` es el argumento del comando `\~` que le coloca el acento ondular. Si sustituimos `3` en vez de `n`, `\~3` producirá $\tilde{3}$: se cambia el argumento pero el comando es el mismo.

2.3.3. Comandos autónomos.

Otros comandos tienen un efecto limitado pero no necesitan argumentos para actuar. Así, `\$3` produce $3, pero `\$` sólo produce $; la 3 no es un "argumento" de `\$`, sino simplemente es el carácter que sigue. De igual modo, `\quad` produce un cuadratín de espacio, sin tomar en cuenta lo que sigue.

2.4 Parámetros de Plain TEX

No todas las palabras de control son comandos; algunas son *parámetros de formato*, es decir, sinónimos abstractos para ciertas cantidades numéricas que TEX usa para sus procesos internos. Por ejemplo, el parámetro `\pageno` denota el número de la página en curso; `\hsize` denota la anchura del texto en la página, y `\vsize` denota la extensión vertical de este texto. Los parámetros se pueden modificar según el esquema

`\parametro =` ⟨*nuevo valor*⟩

donde '`\parametro`' aquí denota cualquier palabra de control con la función de parámetro. (El signo `=` es opcional y puede omitirse; pero es preferible mantenerlo para recordar que se trata de la *asignación* de un nuevo valor al parámetro.)

Por ejemplo, para levantar una parte de un documento con páginas de 16.5 cm × 21 cm, que empieza en la página 57, se puede colocar

```
\pageno=57  \hsize=16.5cm  \vsize=21cm
```

al inicio del archivo fuente.

2.5 Palabras claves

Hay una pequeña colección de palabras especiales, que no son palabras de control (y por ende *no empiezan con vara* \), que TEX reconoce en ocasiones especiales. Por ejemplo, si es necesario declarar una longitud, como 3 cm, por ejemplo, TEX reconocerá la "palabra" `cm` como sinónimo de "centímetro". (Así, se puede teclear `\hsize=16.5cm` para asignar el valor "dieciséis centímetros y medio" al parámetro `\hsize`.) Cuando no se espera una unidad de longitud, el tecleo de `cm` imprime una palabra de dos letras: cm. Las cuatro funciones especiales que demandan estas palabras claves son: medidas de longitud, confección de cajas, confección de goma y operaciones aritméticas internas. La lista completa de palabras claves es:

Función	*Palabras Claves*
Aritmética:	`at`, `by`, `scaled`, `true`
Medidas:	`bp`, `cc`, `cm`, `dd`, `em`, `ex`, `in`, `mm`, `mu`, `pc`, `pt`, `sp`
Cajas:	`depth`, `height`, `spread`, `to`, `width`
Goma:	`fil`, `fill`, `filll`, `minus`, `plus`

Figura 2.2. Palabras claves reconocidos por TEX

Eventualmente veremos el uso de estas palabras claves. Por ahora es suficiente recordar que (a) no llevan vara inicial; (b) sólo son reconocidos como claves en contextos especiales.

2.6 Agrupamiento

Un archivo fuente de TEX es un archivo de texto con comandos intercalados; pero es frecuente encontrar casos en donde uno quiere *limitar el alcance* de algunos comandos. Para poder hacer ésto, TEX usa las *llaves* `{ }` para delimitar bloques de texto a los cuales estos comandos se aplican. Por ejemplo, el comando `\bf` efectúa un cambio de estilo a letra **negrilla**. En el código

```
una palabra en {\bf negrilla} entre palabras claras
```

las llaves limitan el efecto del cambio de estilo a la palabra `negrilla`. Los cambios ambientales efectuados dentro de las llaves se pierden al salir del *grupo* o bloque que ellas delimitan. En este sentido, rodear un bloque de texto con llaves es equivalente a la operación de pantalla de *seleccionar* el bloque con el ratoncillo, con la diferencia importante que la demarcación con `{ }` es permanente, pues queda grabado en el archivo.

⟦ Es importante distinguir entre las *llaves* `{ }`, los *corchetes* `[ ]` y los *paréntesis* `( )`; sólo las llaves poseen la función de agrupamiento en Plain TEX. ⟧

Es a veces incómodo que las palabras de control, como `\TeX`, comen los espacios en blanco que los siguen. Una manera de evitar eso es la de incluir el comando entre llaves: `{\TeX}`, ya que la llave derecha, por no ser letra, termina el comando, y los espacios en blanco que siguen no son ignorados.

Hay otros comandos que *actúan sobre un bloque de texto*. Entonces ese bloque debe indicarse explícitamente, colocándolo entre llaves. Por ejemplo,

```
\centerline{Esta frase debe centrarse.}
```

tiene el efecto de colocar "Esta frase debe centrarse." en un renglón aparte en forma centrada; las llaves delimitan el argumento de `\centerline`, que es el bloque de texto por destacar. Puede colocarse un bloque dentro de otro, así:

```
\centerline{Esta frase debe {\it centrarse}.}
```

Ahora se centra la línea, con la última palabra en letra cursiva:

Esta frase debe *centrarse*.

Las llaves no aparecen en la impresión. Obsérvese que el cambio ambiental `\it` ("letra itálica") se incluye *dentro* del grupo `{\it centrarse}` para localizar su efecto, pero que `\centerline`, un "verbo transitivo", es externo al grupo que afecta.

2.7 Un archivo fuente ejemplar

Consideremos ahora un pequeño ejemplo de un trabajo en Plain TEX.

```
% Un ejemplo de archivo fuente para TeX
\input option_keys
\hrule
\vskip 2cm
\line{Izquierda \hfill Derecha}
\vskip 1.5cm
\centerline{\bf Título Centrado}  \medskip
\centerline{\sl El Autor}          \bigskip

En un archivo fuente, la sangría es automática.  Se puede
teclear las líneas del archivo fuente  en  una    forma
perfectamente    desordenada,  aunque es mala práctica hacer eso
    a propósito. Un espacio o unos veinte espacios separan palabras
de igual manera.

Un párrafo nuevo se obtiene al teclear una línea en blanco.

\medskip
\hrule
\bye
```

Supóngase (en un exceso de realismo) que usted dispone del programa *Textures* para Macintosh, el más "user-friendly" de todas las implementaciones de TEX. Abra el ícono 'Textures™' y teclee este ejemplo (tal cual) en la ventana 'untitled' que se abre automáticamente. Guarde el archivo con algún nombre ('`Tonto`', digamos) y elige la función *Typeset* del menú también llamado '*Typeset*'. El directorio *TEX inputs*, o el mismo directorio donde se guarda el archivo '`Tonto`', debe incluir previamente el archivo '`option_keys`' que siempre acompaña al programa *Textures*. Se abre una nueva ventana TeX log y allí aparece, de inmediato, unos mensajes como los siguientes:

```
Textures 2.0.2 (preloaded format=plain 96.6.7)  24 AUG 1998 20:29
(Tonto (option_keys) [1]
Output written on Tonto (1 page, 833 bytes).
elapsed seconds = 0.28
```

El archivo fuente contiene dos párrafos cortos de texto (separados por una línea en blanco) y varios comandos antes y después para formatear el texto. La primera línea empieza con el carácter `%` que "esconde" todo a su derecha: esta línea es un comentario. La segunda línea lee un archivo externo `macteclado` que contiene los convenios para el tecleo de acentos en español. (Esto es necesario ya que otros sistemas operativos, como MS-DOS, organizan los caracteres acentuados en una forma distinta.) Los comandos `\vskip`, `\medskip` y `\bigskip` solicitan separación vertical de bloques de texto. Los `\hrule` inicial y final trazan reglas horizontales en la página, cuya anchura es la del área de texto en esa página. Los `\centerline` producen renglones centrados, el primero en letra negrilla (obtenida por `\bf`) y el segundo en letra oblicua (debido a `\sl`). La primera línea escrita es un `\line`, que extiende a lo ancho del área de texto escrito, y contiene un *resorte* `\hfill` que se estira para empujar las dos partes de esta línea contra los márgenes izquierda y derecha.

El último comando del ejemplo es `\bye`, que indica a TEX que debe terminar el levantamiento. Cada archivo fuente de Plain TEX debe terminar con `\bye` o alguna instrucción equivalente.

De ahora en adelante veremos, de modo más o menos sistemático, los diversos comandos legales que se pueden incluir en un archivo fuente y los efectos que cada uno produce. La *estructura* de un documento que TEX debe levantar es esencialmente la misma de nuestro pequeño ejemplo: primero hay unos comentarios y comandos preliminares, sigue el texto del documento con algunos comandos de formateo intercalados, y se termina el archivo fuente con el comando `\bye`.

3 Tipos y tamaños de letra

3.1 Tipos Computer Modern

Los caracteres tipográficos existen en diversas formas, con varios pesos, estilos y tamaños. Llamamos *familia tipográfica* a un juego de letras regido por un mismo principio de diseño.

La familia que más se usa hoy día es el Times, originalmente diseñado para el gran periódico The Times de Londres, con el objeto de proporcionar gran legibilidad en una alta densidad. Otra familia muy popular es Helvética, un tipo sanserif: sus caracteres carecen de los pequeños adornos en las esquinas, llamados serifes, que ayudan al ojo a distinguir las letras. Estas familias hoy en día están incorporadas a PostScript, un lenguaje de descripción de página que utilizan las impresoras de laser para imprimir documentos.

La familia *Computer Modern* usada por TEX es de origen distinto. Es una variante digital de los tipos "Modernos" creados por Giambattista Bodoni y Firmin Didot en el siglo XVIII, y su predecesor inmediato es la familia Monotype Modern 8A usado por Addison–Wesley y otros hasta los años sesenta, cuando expiró la tecnología de tipos de plomo caliente. La familia Computer Modern fue creada por METAFONT, un lenguaje de diseño de tipos expresamente compatible con TEX.

Cada familia posee varios *estilos*: romano, cursivo, negrilla, etc. La familia Computer Modern posee los estilos romano, *itálico*, *oblicuo*, **negrilla**, `tipo de teclado`, sanserif, *sanserif oblicuo*, VERSALITA y dos estilos especiales para símbolos matemáticos. Plain TEX no emplea directamente los estilos sanserif; los estilos de texto que usa son romano, itálico, oblicuo, negrilla y tipo de teclado.

En cada estilo hay varias fuentes. Una **fuente** es un juego de caracteres de un estilo y tamaño preciso. Por ejemplo, el estilo `cmr` (Computer Modern Roman) viene en tamaños de 5, 6, 7, 8, 9, 10, 12 y 17 puntos: las fuentes correspondientes son `cmr5`, `cmr6`, ..., `cmr17`. Algunos tamaños más grandes se obtienen por un proceso de "magnificación" de los tipos.

3.2 Cambios de Estilo

Los estilos de Plain TEX son cinco: romano, itálico, oblicuo ("slanted", en inglés), negrilla ("boldface", en inglés) y tipo de teclado. Estos cinco comandos conmutan entre un estilo y otro:

Comando	*Se cambia al estilo:*
`\rm`	romano
`\it`	*itálico*
`\sl`	*oblicuo*
`\bf`	**negrilla**
`\tt`	`tipo de teclado`

Figura 3.1. Comandos que efectúan cambios de estilo

3.2.1. La corrección itálica.

Hay un detalle de calidad importante que debe acompañar el uso de tipos oblicuos o cursivos: después de una palabra oblicua o cursiva que será seguida por una letra recta (romano, negrilla, o tipo de teclado) se debe insertar un pequeño espacio (en impresión) para "enderezar" la letra; este espacio se llama la "corrección itálica" y se obtiene mediante el comando `\/` (vara, raya oblicua)—un "símbolo de control". La regla es:

> Un grupo de palabras que empieza con `{\it␣` o con `{\sl␣` (itálicas u oblicuas) debe terminar con `\/}` en lugar de `}` solamente. Si el grupo termina con puntuación, *se suprime* la corrección itálica *ante una coma o ante un punto*, pero no ante un punto-y-coma, dos puntos, o signos de admiración o interrogación. *Espero* que eso quede *bien claro*.

```
{\it Espero\/} que eso quede {\it bien claro}.
```

3.3 Estilos matemáticos

Los caracteres que se usan en fórmulas matemáticas también vienen en varias fuentes de Computer Modern. Las *letras* en matemáticas representan *variables*: a, b, x, y, f, etc., que normalmente *no* se juntan para formar palabras. Es usual usar para estas variables un estilo *itálico* con algunas diferencias con el *itálico* usual de texto. Este estilo está en archivo bajo la consigna `cmmi`, y lo llamaremos *matitálico*.

Existen también dos fuentes de símbolos: `cmsy` denota una colección de símbolos matemáticos, como $\oplus$, $\rightarrow$, ∇, ∞. Esta fuente también incluye un abecedario de letras caligráficas mayúsculas: $\mathcal{A}$, $\mathcal{B}$, $\ldots$, $\mathcal{Z}$ son símbolos matemáticos de este alfabeto. Finalmente, hay una fuente de símbolos grandes, como $\sum$, $\int$, $\bigotimes$, etc.: esta fuente es `cmex`.

En Plain TEX, las fuentes `cmmi` y `cmsy` aparecen en tamaños de 10, 7 y 5 puntos; los tamaños menores se emplean en exponentes y subíndices, y en fracciones como $\frac{a}{b}$. Los símbolos grandes ocupan una sola fuente, `cmex10`.

Para usar símbolos matemáticos, es necesario encerrarlos entre símbolos de dólar: `$xy$` produce xy. El signo `$` es un carácter especial, cuya función es de *encender y apagar* el ambiente de símbolos matemáticos. Por ejemplo, `x + y` produce x + y (tres caracteres en letra romana), mientras `$x + y$` produce $x+y$, una *expresión matemática* que representa la suma de las dos variables x, y. No sólo aparecen las variables en la fuente "matitálica", sino que el signo + adquiere un espaciamiento apropiado a su alrededor. Estos "efectos matemáticos" comienzan después del primer signo de dólar `$` tecleado; con el segundo `$`, se regresa a texto ordinario. Si se omitiera cualquiera de los dos `$`, se podría producir un efecto tipográfico no deseado. Para prevenir eso, TEX insiste que ciertos comandos estén reservados para el "modo matemático" que ocurre entre los signos de dólar.

4 Fórmulas matemáticas

TEX arma las fórmulas matemáticas de modo que sean fáciles de teclear. Una fórmula complicada es usualmente un juego de subfórmulas menos complicadas conectadas de manera simple; cada subfórmula tiene componentes menos complicados aún. En otras palabras, lo que hay que saber es: (i) cómo teclear fórmulas simples; (ii) cómo combinar subfórmulas.

La fórmula más simple de todas es una letra, como x, que se obtiene al teclear `$x$`. Recuerde que el carácter `$` es un "paréntesis matemático": lo que está entre dos signos `$` está en un "modo matemático", *en donde las reglas de impresión son diferentes* de las reglas para texto corriente.

Por ejemplo, `$x$` da una letra matitálica x, mientras `$2$` da el dígito romano 2: esto es un convenio establecido para fórmulas matemáticas, incorporado automáticamente en TEX. Las teclas de puntuación producen caracteres romanos en impresión, salvo el guión `$-$` que produce un signo menos $-$.

Es un error, casi inevitable al teclear un documento largo, omitir un símbolo `$` de vez en cuando; cuando esto ocurre, TEX tendrá que interpretar texto como fórmulas y viceversa, y probablemente protestará contra ese abuso. Este problema disminuye con la práctica.

Dos símbolos de dólar seguidos, `$$`, encienden y apagan el *modo matemático de despliegue*, en donde una fórmula se destaca en un renglón aparte para mayor legibilidad.

Las reglas de espaciamiento en fórmulas matemáticas son complejas y rígidas: es mejor dejar este trabajo en las manos competentes de TEX. Por lo tanto, se establece un convenio especial dentro del "modo matemático": *TEX ignora todos los espacios tecleados mientras esté en el modo matemático.* Así, si se teclea `$(x + y)/(x - y)$` o `$(x+y)/(x-y)$` o bien `$(x+y) / (x-y)$`, el resultado $(x + y)/(x - y)$ es exactamente el mismo. Por supuesto, se requieren algunos espacios tecleados para terminar palabras de control; pero aparte de eso, el tecleo de espacios dentro de una fórmula puede hacerse en una forma totalmente arbitraria. Se aconseja intercalar suficientes espacios para que el tecleo de la fórmula sea fácilmente legible; por ejemplo, es un buen hábito rodear *verbos* como '`=`' con espacios en blanco.

4.1 El bestiario de símbolos

4.1.1. Caracteres matemáticos.

Las letras (de `A` a `Z` y de `a` a `z`) representan "variables" en ecuaciones, etcétera, y se representan con letras *matitálicas*. Los dígitos de `0` a `9` y los caracteres de puntuación `! ? . | / ‘ @ "` se imprimen con letras romanas. Todos estos caracteres se imprimen seguidos, sin colocar espacio extra a su alrededor.

4.1.2. Letras griegas.

En el modo matemático, se dispone de un gran surtido de símbolos especiales que normalmente no se ocupan en texto corriente. En primer lugar, tenemos *el alfabeto griego*: si se teclea `$\alpha,\beta,\gamma$`, se obtiene α, β, γ. Las mayúsculas requieren comandos con letra inicial mayúscula: `$\Gamma,\Delta,\gamma,\delta$` produce $\Gamma, \Delta, \gamma, \delta$. He aquí la lista:

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

Figura 4.1. Letras griegas minúsculas en fórmulas

El comando `\omicron` no hace falta, pues se usa la letra matitálica o; algunas letras mayúsculas son iguales a ciertas mayúsculas romanas y los otros se obtienen por comandos cuyas letras iniciales son mayúsculas:

A	`{\rm A}`	H	`{\rm H}`	N	`{\rm N}`	T	`{\rm T}`
B	`{\rm B}`	Θ	`\Theta`	Ξ	`\Xi`	Υ	`\Upsilon`
Γ	`\Gamma`	I	`{\rm I}`	O	`{\rm O}`	Φ	`\Phi`
Δ	`\Delta`	K	`{\rm K}`	Π	`\Pi`	X	`{\rm X}`
E	`{\rm E}`	Λ	`\Lambda`	P	`{\rm P}`	Ψ	`\Psi`
Z	`{\rm Z}`	M	`{\rm M}`	Σ	`\Sigma`	Ω	`\Omega`

Figura 4.2. Letras griegas mayúsculas en fórmulas

4.1.3. Símbolos ordinarios.

Además de las letras griegas, TEX usa muchos otros símbolos en sus fórmulas, como $\leq$, $\in$, ∞, $\approx$, $\otimes$, que se obtienen con comandos como `$\leq, \in, \infty, \approx, \otimes$`. Estos símbolos se clasifican según su significado en fórmulas. Algunos de los símbolos tienen más de un nombre; se puede emplear cualquiera de las alternativas dadas. (Si los nombres sugeridos por Plain TEX no son de su gusto, usted los puede cambiar, como veremos en la § 5).

La lista siguiente consta de símbolos que se emplean como variables ordinarias, y no tienen atributos especiales de espaciamiento.

$\aleph$	`\aleph`	∂	`\partial`	$\vert$	`|`, `\vert`	$\flat$	`\flat`
$\hbar$	`\hbar`	∞	`\infty`	$\Vert$	`\\|`, `\Vert`	$\natural$	`\natural`
$\imath$	`\imath`	$\prime$	`\prime`	$\backslash$	`\backslash`	$\sharp$	`\sharp`
$\jmath$	`\jmath`	$\emptyset$	`\emptyset`	$\angle$	`\angle`	$\clubsuit$	`\clubsuit`
ℓ	`\ell`	∇	`\nabla`	$\triangle$	`\triangle`	$\diamondsuit$	`\diamondsuit`
$\wp$	`\wp`	$\surd$	`\surd`	$\forall$	`\forall`	$\heartsuit$	`\heartsuit`
$\Re$	`\Re`	$\top$	`\top`	$\exists$	`\exists`	$\spadesuit$	`\spadesuit`
$\Im$	`\Im`	$\bot$	`\bot`	$\neg$	`\neg`, `\lnot`		

Figura 4.3. Símbolos matemáticos ordinarios

4.1.4. Operaciones binarias.

Los caracteres `+`, `-`, `*` representan *operaciones binarias* matemáticas (`+` es sumar, `-` es restar) que ligan dos cosas: la suma `$2+3$` expresa que 2 y 3 se combinan de una cierta forma (es irrelevante que el resultado sea 5) y se imprime 2+3 con un poco de espacio alrededor del signo +. Otras operaciones se producen con comandos: `$2 \times 3$`, `$8 \div 4$`, `$n \pm 1$`, `$A \cap B$` producen 2×3, $8 \div 4$, $n \pm 1$, $A \cap B$. Aquí está la lista completa de operaciones binarias:

$+$	`+`	$-$	`-`	$*$	`*`, `\ast`	$\star$	`\star`
$\pm$	`\pm`	$\cap$	`\cap`	$\odot$	`\odot`	$\dagger$	`\dagger`
$\mp$	`\mp`	$\cup$	`\cup`	$\otimes$	`\otimes`	$\ddagger$	`\ddagger`
$\setminus$	`\setminus`	$\uplus$	`\uplus`	$\oplus$	`\oplus`	$\wr$	`\wr`
$\cdot$	`\cdot`	$\sqcap$	`\sqcap`	$\ominus$	`\ominus`	$\triangleleft$	`\triangleleft`
$\times$	`\times`	$\sqcup$	`\sqcup`	$\oslash$	`\oslash`	$\triangleright$	`\triangleright`
$\div$	`\div`	$\diamond$	`\diamond`	$\bigcirc$	`\bigcirc`	$\bigtriangleup$	`\bigtriangleup`
$\vee$	`\vee`, `\lor`	$\circ$	`\circ`	$\amalg$	`\amalg`	$\bigtriangledown$	`\bigtriangledown`
$\wedge$	`\wedge`, `\land`	$\bullet$	`\bullet`				

Figura 4.4. Operaciones binarias en matemáticas

4.1.5. Relaciones.

Los caracteres `=`, `<`, `>`, `:` expresan *relaciones* entre lo que viene antes y lo que viene después; `$2 < 3$` significa "2 es menor que 3" y se imprime $2 < 3$, con espacio alrededor del signo <. El espacio que rodea una "relación" matemática es levemente mayor que el espacio alrededor de una "operación". Otras relaciones se producen con comandos: `$2 \leq 3$`, `$5 \geq 4$`, `$x \equiv y$`, `$x \in A$`, `$A \subset B$`, `$X \to Y$`, `$a \mapsto B$` producen $2 \leq 3$, $5 \geq 4$, $x \equiv y$, $x \in A$, $A \subset B$, $X \to Y$, $a \mapsto B$. La lista completa de relaciones aparece abajo:

$<$	`<`	$>$	`>`	$=$	`=`	$:$	`:`
$\not<$	`\not<`	$\not>$	`\not>`	$\neq$	`\not=`, `\neq`, `\ne`	$\notin$	`\notin`
$\leq$	`\leq`, `\le`	$\geq$	`\geq`, `\ge`	$\equiv$	`\equiv`	$\in$	`\in`
$\ll$	`\ll`	$\gg$	`\gg`	$\doteq$	`\doteq`	$\ni$	`\ni`, `\owns`
$\prec$	`\prec`	$\succ$	`\succ`	$\sim$	`\sim`	$\approx$	`\approx`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\simeq$	`\simeq`	$\cong$	`\cong`
$\subset$	`\subset`	$\supset$	`\supset`	$\mid$	`\mid`	$\asymp$	`\asymp`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\parallel$	`\parallel`	$\bowtie$	`\bowtie`
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\smile$	`\smile`	$\frown$	`\frown`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$\models$	`\models`	$\perp$	`\perp`
$\propto$	`\propto`						

Figura 4.5. Relaciones matemáticas

Los comandos `\mid` y `\parallel` producen los mismos símbolos que `|` y `\|`, pero con espaciamiento apropiado para relaciones. Es importante prestar atención a estas pequeñas diferencias.

Otras relaciones compuestas se obtienen al teclear dos relaciones seguidas; estos forman una sola unidad que actúa como relación. Por ejemplo, $f(x) := x + 2$ se obtiene simplemente al teclear `$f(x) := x + 2$`; fíjese en la relación compuesta `:=` (asignación).

En particular, el comando `\not` produce una raya oblicua / *con un retroceso* a la posición inicial; si otra relación lo sigue, el efecto tipográfico es de tachar o "negar" la segunda relación. Por ejemplo, `\not=` produce el símbolo de desigualdad $\neq$ (que también se obtiene con `\neq` o `\ne`), `\not\equiv` da $\not\equiv$, `\not\subset` da $\not\subset$, etc. Para negar $\in$ (`\in`), se dispone de `\notin`, que es un solo comando.

4.1.6. Flechas.

Una clase particular de relaciones, que abundan en escritos matemáticos, son las flechas. TEX tiene un gran surtido de flechas horizontales y verticales:

$\leftarrow$	`\gets`, `\leftarrow`	$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\Uparrow`
$\rightarrow$	`\to`, `\rightarrow`	$\longrightarrow$	`\longrightarrow`	$\downarrow$	`\downarrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Downarrow$	`\Downarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`	$\updownarrow$	`\updownarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`	$\nearrow$	`\nearrow`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`	$\searrow$	`\searrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`	$\swarrow$	`\swarrow`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`	$\nwarrow$	`\nwarrow`
$\rightleftharpoons$	`\rightleftharpoons`	$\iff$	`\iff`		

Figura 4.6. Flechas en matemáticas

El símbolo `\iff` ("si y solo si") es un `\Longleftrightarrow` con aún más espacio alrededor, que es lo que se requiere para indicar una relación de equivalencia lógica entre dos afirmaciones simbólicas.

4.1.7. Puntuación.

Los caracteres `,` y `;` sirven para puntuar listas de variables dentro de fórmulas: hay un poco de espacio después de ellos, pero nada antes: `$(x,y;z)$` produce $(x, y; z)$. El apóstrofe `$'$` produce una prima $'$, que es una versión más pequeña del símbolo `\prime`, colocado en la posición de un exponente. A veces se requieren dos puntos con las mismas reglas de espaciamiento que un punto y coma; la fórmula común $f\colon A \to B$ tiene dos puntos como signo de puntuación y no como relación. El comando `\colon` produce este signo de puntuación: `$f\colon A \to B$`. (Obsérvese que el código `$f: A \to B$` produce la fórmula incorrecta $f : A \to B$, ya que no debe haber espacio entre la f y los dos puntos.)

4.1.8. Delimitadores.

Las subfórmulas se distinguen visualmente encerrándolos entre paréntesis, corchetes, llaves visibles, etcétera, los cuales se llaman colectivamente "delimitadores". Algunos se colocan a la izquierda de la subfórmula: `(`, `[`, `\{` y `\langle` producen (, [, { y $\langle$ en modo matemático. Otros se colocan a la derecha: `)`, `]`, `\}` y `\rangle` producen),], } y $\rangle$. A veces se requieren delimitadores en el centro: `/` y `\backslash` producen / y \ (el `\backslash` solo funciona en modo matemático). Finalmente, hay dos delimitadores verticales: `|` y `\|` producen | y ‖, que pueden emplearse en cualquier posición. Las flechas verticales y algunos otros símbolos funcionan también como delimitadores; he aquí el catálogo completo:

(	`(`	)	`)`	[	`[`, `\lbrack`	]	`]`, `\rbrack`
$\vert$	`|`, `\vert`	$\Vert$	`|`, `\Vert`	{	`\{`, `\lbrace`	}	`\}`, `\rbrace`
/	`/`	\	`\backslash`	$\langle$	`\langle`	$\rangle$	`\rangle`
$\lfloor$	`\lfloor`	$\rfloor$	`\rfloor`	$\lceil$	`\lceil`	$\rceil$	`\rceil`
$\uparrow$	`\uparrow`	$\downarrow$	`\downarrow`	$\updownarrow$	`\updownarrow`		
$\Uparrow$	`\Uparrow`	$\Downarrow$	`\Downarrow`	$\Updownarrow$	`\Updownarrow`		
$\lgroup$	`\lgroup`	$\rgroup$	`\rgroup`	$\lmoustache$	`\lmoustache`	$\rmoustache$	`\rmoustache`

Figura 4.7. Delimitadores en fórmulas matemáticas

Los últimos cuatro delimitadores se obtienen solamente en tamaños grandes, inadecuados para uso dentro de párrafos pero útiles para fórmulas desplegadas. Son de uso poco frecuente. Los "corchetes angulares" `$\langle$` y `$\rangle$`, que producen $\langle$ y $\rangle$, no deben confundirse con las relaciones $<$ y $>$.

4.1.9. Operadores grandes.

En matemáticas, ocurren muchas *sumas* e *integrales*, que se escriben con símbolos especiales: $\sum$ para sumas, $\int$ para integrales. Para teclearlos, solo hay que saber que `\sum` produce $\sum$ y que `\int` produce $\int$. En despliegues, estos símbolos tienen un tamaño mayor de lo que tienen en texto; por eso, se llaman "operadores grandes".

La lista completa de operadores grandes es la que sigue:

$\sum$	$\sum$	`\sum`	$\bigcap$	$\bigcap$	`\bigcap`	$\odot$	$\bigodot$	`\bigodot`
$\prod$	$\prod$	`\prod`	$\bigcup$	$\bigcup$	`\bigcup`	$\otimes$	$\bigotimes$	`\bigotimes`
$\coprod$	$\coprod$	`\coprod`	$\bigsqcup$	$\bigsqcup$	`\bigsqcup`	$\oplus$	$\bigoplus$	`\bigoplus`
$\int$	$\int$	`\int`	$\bigvee$	$\bigvee$	`\bigvee`	$\uplus$	$\biguplus$	`\biguplus`
$\oint$	$\oint$	`\oint`	$\bigwedge$	$\bigwedge$	`\bigwedge`	$\smallint$	$\smallint$	`\smallint`

Figura 4.8. Operadores grandes en tamaños de texto y de despliegue

El comando `\smallint` es de mediana estatura solamente y es de uso poco frecuente. Hay que distinguir entre `\sum` y `\Sigma`, entre `\prod` y `\Pi`, y entre `\coprod` y `\amalg`.

Todos estos símbolos se permiten solamente en el modo matemático, pero no en texto corriente; esto es así para controlar la colocación de signos `$`. Si TEX encuentra un comando de símbolo matemático en texto ordinario, emite una protesta y inserta un signo `$` por propia iniciativa para tratar de arreglar las cuentas. (Supondrá, pues, que un signo `$` ha sido omitido por descuido.)

4.2 Raíces y barras

Los macros `\sqrt`, `\underline` y `\overline` actúan sobre subfórmulas. Colocan barras en cima o por debajo de lo que sigue, y `\sqrt` además coloca un surdo $\surd$ adelante para indicar que se trata de una raíz cuadrada. (Para raíces cúbicas y otras, se usa el par inseparable de comandos `\root...\of`.) Por ejemplo:

`\sqrt 2`	$\sqrt{2}$	`\sqrt{x+3}`	$\sqrt{x+3}$
`\underline 4`	$\underline{4}$	`\overline{x+y}`	$\overline{x+y}$
`\underline{x^2+y^2}`	$\underline{x^2+y^2}$	`\root 3 \of 4`	$\sqrt[3]{4}$

También se pueden colocar *flechas* sobre una subfórmula. Los comandos `\overrightarrow` y `\overleftarrow` producen flechas superiores, aptas para vectores:

`$\overrightarrow M = \overrightarrow{AB},$`	$\overrightarrow{M} = \overrightarrow{AB},$
`$\overleftarrow N = \overleftarrow{CD}.$`	$\overleftarrow{N} = \overleftarrow{CD}.$

4.3 Subíndices y exponentes

Los subíndices y exponentes se producen, en modo matemático *solamente*, con los caracteres reservados `_` (para subíndices) y `^` (para exponentes). Estos rebajan o elevan el caracter inmediatamente siguiente: `$x_3$` produce x_3, `$x^2$` produce x^2. Además, `$x^2y^2$` produce x^2y^2. Para rebajar o elevar dos o más caracteres, hay que colocar el subíndice o exponente entre llaves: `$x^{2y}$` produce x^{2y}.

Resulta que x^{y^z} no significa lo mismo que x^{yz} en fórmulas matemáticas: debido a eso, se prohibe teclear `$x^y^z$` por ser ésta una *expresión ambigua*. TEX protestará sobre "exponentes dobles" si encuentra `x^y^z` en modo matemático, y sobre "subíndices dobles" si encuentra algo como `a_b_c`. Hay que *usar grupos* para distinguir los casos: `$x^{y^z}$` o `$x^{yz}$`; `$a_{b_c}$` o `$a_{bc}$`.

Si se teclea `$(x^2)^3 = {(x^2)}^3$`, se obtiene $(x^2)^3 = {(x^2)}^3$. Obsérvese las diferentes posiciones del exponente 3: en el primer caso se coloca la 3 como exponente al paréntesis `)`, y en el segundo es un exponente a la subfórmula `(x^2)` como un todo. La primera opción es tipográficamente preferible, y además elimina la necesidad de teclear muchos pares de llaves.

Para teclear *un exponente y un subíndice* simultáneamente, no hay ambigüedad: `$x_k^2$` y `$x^2_k$` ambos producen x_k^2, con la k directamente debajo de la 2. Es preferible, como regla de estilo, teclear el subíndice *antes* del exponente, es decir, obtener x_k^2 con `$x_k^2$`, pero se permite invertir ese orden.

Un exponente que ocurre mucho en matemáticas es la "prima" $'$, que se obtiene con el comando `\prime`; así, `$x^\prime$` produce x', `$y^{\prime\prime}$` produce y'', etcétera. Para mayor comodidad, Plain TEX permite el uso del *apóstrofe* `'` en modo matemático como sinónimo de `\prime`; entonces se pueden obtener x' y y'' con `$x'$` y `$y''$` simplemente.

4.4 Acentos en modo matemático

Los acentos para texto, como `\'`, `\‘`, `\"` etcétera, no son permitidos en modo matemático. Pero poseen equivalentes que se usan *solo en modo matemático*. La lista es:

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

Figura 4.9. Acentos en fórmulas matemáticas

Estos acentos se colocan rutinariamente sobre cualquier letra (no necesariamente una vocal) o inclusive sobre otros símbolos: `$\dot\times$` produce $\dot\times$. El último de la lista, la flechita `\vec` (que se usa para denotar vectores), no tiene equivalente en texto.

El acento ondular `\tilde` y el techo `\hat` tienen versiones mas grandes:

`$\widehat x = \widehat{yz},$` $\widehat x = \widehat{yz},$

`$\widetilde{xy} = \widetilde z.$` $\widetilde{xy} = \widetilde z.$

Las flechas superiores `\overrightarrow` y `\overleftarrow` pueden considerarse "acentos sagitarios", que se extienden lo necesario para cubrir su argumento; sus tamaños mínimos son los de las flechas `\to` y `\gets`.

`$\overrightarrow{AB} = \overrightarrow{x+y+z}$` $\overrightarrow{AB} = \overrightarrow{x+y+z}$

En general, se recomienda usar estas flechas largas y `\overline` con letras mayúsculas o con más de un símbolo, reservando `\vec` y `\bar` para letras minúsculas.

`$\bar z = \overline z = \overline P$` $\bar z = \overline z = \overline P$

`$\vec x = \overrightarrow x = \overrightarrow T$` $\vec x = \overrightarrow x = \overrightarrow T$

Para acentuar i, j en modo matemático, hay que reemplazar `i`, `j` por `\imath`, `\jmath` que producen $\imath$, $\jmath$ sin puntos en modo matemático. Así, $\hat\imath$ se obtiene con `$\hat\imath$`.

4.5 Fórmulas desplegadas

Un *despliegue* es una fórmula colocada en un renglón aparte para destacarla visualmente y separarla del texto del párrafo donde ocurre. Por ejemplo,

$$\int_0^1 \int_0^1 \frac{dy\,dx}{1-xy} = \sum_{n=0}^{\infty} \int_0^1 \int_0^1 x^n y^n \,dy\,dx = \sum_{n=0}^{\infty} \biggl(\int_0^1 t^n \,dt \biggr)^2 = \sum_{n=0}^{\infty} \frac{1}{(n+1)^2} = \frac{\pi^2}{6}$$

es una fórmula demasiado grande y compleja como para esconderla dentro de un párrafo de texto. Un despliegue empieza y termina con `$$`; Para destacarlos en el archivo fuente, se recomienda teclear `$$` siempre en una línea aparte. Por ejemplo, se obtuvo el último despliegue al teclear:

```
$$
\int_0^1 \int_0^1 \frac{dy\,dx}{1 - xy}
 = \sum_{n=0}^\infty \int_0^1 \int_0^1 x^n y^n \,dy\,dx
 = \sum_{n=0}^\infty \biggl( \int_0^1 t^n \,dt \biggr)^2
 = \sum_{n=0}^\infty \frac1{(n+1)^2} = \frac{\pi^2}6
$$
```

Para *enumerar una fórmula* con una "etiqueta" (un número entre paréntesis, digamos) al lado de una fórmula desplegada, teclee `\eqno`⟨*etiqueta*⟩ al final del despliegue, justo antes de los `$$` finales. Todo lo que viene entre el `\eqno` y los `$$` se coloca al lado derecho del despliegue. Para obtener el número al lado izquierdo, teclee `\leqno` en vez de `\eqno`, pero siempre después de la ecuación principal. Por ejemplo:

```
$$
x^2 - y^2 = (x - y)(x + y),  \eqno(42)
$$
mientras
$$
x^3 - y^3 = (x - y)(x^2 + xy + y^2).  \leqno(43)
$$
```

produce el par de despliegues:

$$x^2 - y^2 = (x - y)(x + y), \tag{42}$$

mientras

$$x^3 - y^3 = (x - y)(x^2 + xy + y^2). \tag{43}$$

Se recomienda, para mantener uniformidad, usar solamente `\eqno` o solamente `\leqno` en un documento, sin mezclarlos.

4.6 Fracciones

Es común encontrar en fórmulas matemáticas diversas fracciones y otras cosas con unos símbolos montados sobre otros:

$$\frac{1}{2} \quad \text{y} \quad \frac{n+1}{3} \quad \text{y} \quad \binom{n+1}{3}$$

se obtienen *en despliegues* al teclear `$$1 \over 2$$`, `$$n+1 \over 3$$`, `$$n+1 \choose 3$$`. Si se teclea `$1 \over 2$` y `$n+1 \choose 3$`, se obtienen $\frac{1}{2}$ y $\binom{n+1}{3}$ *en un tamaño menor* dentro del renglón de texto, lo cual dificulta su lectura; por eso, se acostumbra desplegar las fracciones cuando sea posible.

El comando `\over` divide la fórmula en dos partes: la parte que precede el `\over` es el *numerador* de la fracción, y la parte que lo sigue es el *denominador*. Si la fracción no abarca toda la fórmula, *hay que limitar el alcance de* `\over` *con llaves* alrededor de la fracción:

`$$ x+y^2  \over  k+1 $$`	$$x+y^2 \over k+1$$
`$$ {x+y^2 \over k}+1 $$`	$${x+y^2 \over k}+1$$
`$$ x+{y^2 \over k}+1 $$`	$$x+{y^2 \over k}+1$$
`$$ x+{y^2 \over k+1} $$`	$$x+{y^2 \over k+1}$$
`$$ x+y^{2 \over k+1} $$`	$$x+y^{2 \over k+1}$$

Como regla general, es aconsejable encerrar una fracción entre llaves *siempre*; esto evita dificultades sobre el alcance de `\over`.

Al igual que `$a_b_c$` y `$a^b^c$`, la frase `$a \over b \over c$` es ambigua y por lo tanto es prohibida; hay que usar llaves para distinguir las partes de las fracciones:

`$$`		`$$`	
`{{a \over b} \over c}`	$${a \over b} \over c$$	`{a \over {b \over c}}`	$$a \over {b \over c}$$
`$$`		`$$`	

Debido al cambio de tamaños, es mejor usar la raya / para formar fracciones: $\frac{a}{b} = a/b$. El ejemplo anterior se ve mucho mejor de la forma:

```
$$                          $$
{a/b \over c}    a/b/c      {a \over b/c}    a/(b/c)
$$                          $$
```

Hay otros dos comandos que tienen un formato similar a `\over`, y tienen las mismas reglas de alcance: `\atop` ("encima") da una fracción sin barra, y `\choose` da una fracción sin barra encerrada entre paréntesis:

```
$$
{n \atop k+1} \neq {n\choose k}
$$
```

$$\begin{matrix} n \\ k+1 \end{matrix} \neq \binom{n}{k}$$

⟦El nombre `\choose` se usa porque $\binom{5}{2}$ significa el número de maneras de elegir 2 objetos de entre una lista de 5 objetos dados. Resulta que hay 15 formas diferentes de hacer esa elección: $\binom{5}{2} = 15$.⟧

4.6.1. Tamaños de letras en fórmulas desplegadas.

TEX reconoce cuarto "estilos" de fórmulas, que controlan la forma de las letras. Podrían llamarse "estilo de despliegue", "estilo de texto", "estilo de índices" y "estilo de índices de segundo orden". Una fórmula sin subíndices, exponentes ni fracciones usa el estilo de texto para las letras y los símbolos que no son operadores grandes. Hemos notado que los operadores grandes tienen un tamaño mayor en fórmulas desplegadas que en fórmulas dentro de un párrafo de texto; el tamaño mayor corresponde al estilo de despliegue, el menor al estilo de texto. Un subíndice o exponente a una subfórmula en estilo de texto aparece en el estilo de índices, y un subíndice o exponente a una subfórmula en estilo de índices aparece en el estilo de índices de segundo orden. Los tamaños de las fuentes para variables matitálicas que aparecen en los cuatro estilos tienen una proporción aproximada de 10 : 10 : 7 : 5; en Plain TEX, sin magnificación, se usan fuentes de 10 pt, 7 pt y 5 pt para respetar estas proporciones.

Una fracción en una fórmula tiene su numerador y su denominador en el siguiente estilo más pequeño, hasta donde esto sea posible. Por ejemplo, una fracción dentro de texto tendrá su numerador y denominador en el estilo de índices, pero una fracción en un despliegue los tendrá en el estilo de texto. Se puede cambiar el estilo explícitamente con los comandos `\displaystyle`, `\textstyle`, `\scriptstyle` y `\scriptscriptstyle`, que se aceptan en el modo matemático solamente y efectúan cambios ambientales. Compare:

```
$e^{1\over2}$                                  e^{1\over2}
$e^{\scriptstyle1\over\scriptstyle2}$          e^{\scriptstyle1\over\scriptstyle2}
$e^{1/2}$                                      e^{1/2}
```

$e^{\frac{1}{2}}$
$e^{\frac{1}{2}}$
$e^{1/2}$

Aquí e está en el estilo de texto (o de despliegue) y su exponente está en el estilo de índices. Cuando este exponente es `{1\over2}`, las cifras `1` y `2` están en el estilo de índices de segundo orden. Los `\scriptstyle` en el ejemplo dan estas cifras en un tamaño más grande. Sin embargo, en casos como este, se recomienda usar la forma `1/2`, que tiene mejor apariencia tipográfica.

4.7 Funciones matemáticas

Ciertas fórmulas usan abreviaturas como sin, cos, log, lim, etcétera, que forman palabras especiales y aparecen en letras romanas dentro de las fórmulas para distinguirlas visualmente. Estas se obtienen por *comandos* tales como `\sin`, `\cos`, `\log`, `\lim` que funcionan solamente en modo matemático. Lo que hay que recordar es que estas palabritas se teclean con una \ adelante, que las convierte en comandos. El catálogo de comandos de Plain TEX que dan palabras con letra romana

es el siguiente:

```
\arccos  \cos   \csc  \exp  \ker     \limsup  \min  \sinh
\arcsin  \cosh  \deg  \gcd  \lg      \ln       \Pr   \sup
\arctan  \cot   \det  \hom  \lim     \log      \sec  \tan
\arg     \coth  \dim  \inf  \liminf  \max      \sin  \tanh
```

Figura 4.10. Funciones matemáticas con letras romanas

Estas palabras especiales pueden llevar índices mediante `_` y `^`, como por ejemplo:

```
$$
\lim_{x\to0} {\sin^2x \over x^2} = 1
$$
```

$$\lim_{x\to0} {\sin^2x \over x^2} = 1$$

(observe el uso de `\to` para producir la flecha →.)

4.8 Delimitadores

4.8.1. Delimitadores en tamaños `\big` *y* `\bigg`.

Los delimitadores vienen en varios tamaños. A veces un tamaño más grande de paréntesis es apropiado para encerrar una subfórmula que ya incluye otros paréntesis; esto ayuda la distinción visual de la jerarquía de fórmulas. Lo mismo puede suceder con corchetes, barras, y otros delimitadores. Los tamaños más grandes se obtienen al colocar `\bigl` ante un delimitador izquierdo y `\bigr` ante un delimitador derecho. Por ejemplo,

```
$\bigl(x-s(x)\bigr)\bigl[y-s[y]\bigr]$
```

$\bigl(x-s(x)\bigr)\bigl[y-s[y]\bigr]$

```
$\bigl| |x|+|y| \bigr|$
```

$\bigl| |x|+|y| \bigr|$

En despliegues, se pueden colocar `\biggl` y `\biggr` ante los delimitadores respectivos para obtener delimitadores aun más grandes. Esto es particularmente apropiado para encerrar fracciones:

```
$$
\biggl({a+b \over c+d}\biggr)
$$
```

$$\biggl({a+b \over c+d}\biggr)$$

⟦ Hay otros dos tamaños de delimitadores: `\Bigl` y `\Bigr` dan un tamaño intermedio mayor que `\bigl` y `\bigr` pero menor que `\biggl` y `\biggr`; también hay `\Biggl` y `\Biggr`, que son mayores aún que `\biggl` y `\biggr`. Estos pueden emplearse en despliegues, pero son de poco uso. Los delimitadores `\lgroup`, `\rgroup`, `\lmoustache`, `\rmoustache` se obtienen solamente en los tamaños `\Big`, `\bigg` o mayor. ⟧

4.8.2. Los comandos `\left` *y* `\right`.

Otro método de encerrar una fórmula con delimitadores es el de colocar `\left` ante el delimitador izquierdo, y `\right` ante el delimitador derecho. Con esta opción, TEX tome nota de la subfórmula entre el `\left`⟨*delimitador*⟩ y el `\right`⟨*delimitador*⟩, calcula rápidamente su extensión vertical, y luego elige *automáticamente* el tamaño mínimo que deben llevar los ⟨*delimitador*⟩ para encerrar la subfórmula intermedia. Por ejemplo, `$\left( a+b \right)$` produce $(a+b)$, que es idéntico al resultado de teclear `$(a+b)$`; pero observe:

```
$$
1+\left( 1 \over 1-x^2 \right)^3
$$
```

$$1+\left(1 \over 1-x^2 \right)^3$$

pues en este caso se eligen `\biggl(` y `\biggr)` para encerrar la fracción.

Hay dos cosas de notar acerca de `\left` y `\right`. En primer lugar, ellos *funcionan como un par de llaves* para demarcar un grupo: la subfórmula `1\over1-x^2` no extiende el alcance del `\over` más allá del `\left(` y del `\right)`. En segundo lugar, *deben aparecer en pareja*: una fórmula que empieza con un `\left`⟨*delimitador*⟩ debe terminar con un `\right`⟨*delimitador*⟩ para que se pueda saber cuál es la subfórmula intermedia. Si se omite un `\right`⟨*delimitador*⟩ después de un `\left`⟨*delimitador*⟩, o si se omite un `\left`⟨*delimitador*⟩ antes de un `\right`⟨*delimitador*⟩, TEX se sentirá ofendido y protestará contra ese abuso.

4.9 Matrices

Una "matriz" es *un arreglo rectangular* de símbolos (dígitos, letras o subfórmulas) que se organizan en varias *filas y columnas*. Las matrices normalmente aparecen en despliegues. Por ejemplo:

```
$$                                                    1  2  3
\matrix{1 & 2 & 3\cr 4 & 5 & 6\cr 7 & 8 & 9\cr}      4  5  6
$$                                                    7  8  9
```

La fórmula

$$\det\begin{bmatrix}1 & 2\\ 0 & 1\end{bmatrix} = \begin{vmatrix}1 & 2\\ 0 & 1\end{vmatrix} = 1$$

se obtiene al teclear

```
$$
\det\left[ \matrix{1 & 2\cr 0 & 1\cr} \right]
  = \left| \matrix{1 & 2\cr 0 & 1\cr} \right| = 1
$$
```

El contenido de la matriz es el argumento (entre llaves) del comando `\matrix`. Las *filas* terminan con `\cr`, que es una especie de "retorno de carro" explícito (↵ sólo da un espacio en blanco). Las columnas se separan con el carácter especial `&`, que en este caso juega el papel de "tabulador" entre columnas. Los delimitadores alrededor de la matriz pueden colocarse con `\left` y `\right`.

Es muy común encerrar una matriz entre paréntesis. En Plain TEX, se usa `\pmatrix{...}` para no tener que teclear `\left(\matrix{...}\right)`. Por ejemplo,

```
$$
\pmatrix{3 & 4\cr 5 & 6\cr}.      $$\begin{pmatrix}3 & 4\\ 5 & 6\end{pmatrix}.$$
$$
```

4.10 Alineamiento de ecuaciones

4.10.1. Alineamientos no numerados.

A veces hay que desplegar más de una línea a la vez. Por ejemplo, si hay una lista de dos o más ecuaciones, deben ser desplegadas de tal modo que los signos de igualdad estén verticalmente *alineados*. Tipográficamente, se trata de una especie de matriz, con filas separadas por `\cr`, y dos columnas (la parte antes del signo = y la parte del signo = en adelante), separadas por un tabulador `&`. El comando apropiado es `\eqalign`, que produce un alineamiento centrado con la primera columna justificada a la derecha y la segunda justificada a la izquierda (no se permiten más de dos columnas). Ejemplo:

$$\begin{aligned}SO(n) &= \{A : A^T A = I, \det A = 1\},\\ U(n) &= \{A : A^* A = I\},\\ SU(n) &= \{A \in U(n) : \det A = 1\},\\ Sp(n) &= \{A \in U(2n) : A^T J A = J\}.\end{aligned}$$

(Obsérvese que aquí los = quedan verticalmente alineados.) Esto resulta de:

```
$$
\eqalign{SO(n) &= \{A : A^T A = I, \det A = 1\}, \cr
          U(n) &= \{A : A^* A = I\}, \cr
         SU(n) &= \{A \in U(n): \det A = 1\}, \cr
         Sp(n) &= \{A \in U(2n): A^T J A = J\}. \cr}
$$
```

Otro ejemplo es:

$$\begin{aligned} E(X) &= \sum_{k=0}^{n} k \binom{n}{k} p^k (1-p)^{n-k} = \sum_{k=1}^{n} k \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \\ &= np \sum_{k=1}^{n} \frac{(n-1)!}{(k-1)!(n-k)!} p^{k-1} (1-p)^{n-k} = np \sum_{r=0}^{n-1} \binom{n-1}{r} p^r (1-p)^{n-1-r} \\ &= np\big(p + (1-p)\big)^{n-1} = np. \end{aligned}$$

Esto se obtiene de:

```
$$
\eqalign{
E(X) &= \sum_{k=0}^n k {n \choose k} p^k (1-p)^{n-k}
      = \sum_{k=1}^n k {n! \over k! (n-k)!} p^k (1-p)^{n-k} \cr
&= np \sum_{k=1}^n {(n-1)! \over (k-1)! (n-k)!} p^{k-1} (1-p)^{n-k}
 = np \sum_{r=0}^{n-1} {n-1 \choose r} p^r (1-p)^{n-1-r} \cr
&= np \bigl( p + (1-p) \bigr)^{n-1} = np. \cr}
$$
```

Obsérvese que en este ejemplo la fórmula abarca tres renglones (separados por `\cr`) pero el segundo y el tercero tienen parte izquierda vacía, y por eso empiezan directamente con el tabulador `&`.

4.10.2. Alineamientos con etiquetas.

Si se quiere colocar etiquetas en un despliegue alineado, cada renglón constará de tres partes: la parte izquierda, la parte derecha, y la etiqueta (que puede colocarse junto a cualquiera de los dos márgenes). Los comandos `\eqalignno` y `\leqalignno` (el segundo coloca las etiquetas a la izquierda) se usan para alinear las tres partes. En cada línea, hay *uno o dos* tabuladores `&`: el primero divide la izquierda y la derecha de la fórmula; si aparece un segundo `&`, todo lo que sigue hasta el `\cr` es la etiqueta para esta línea. Ejemplo:

```
$$
\eqalignno{
1+1 &= 2  &(9) \cr
1+1+1 &= 2+1 = 3  &(9') \cr
1+1+1+1 &= 3+1 = 4  &(9'') \cr}
$$
```

produce:

$$\begin{aligned} 1+1 &= 2 && (9) \\ 1+1+1 &= 2+1 = 3 && (9') \\ 1+1+1+1 &= 3+1 = 4 && (9'') \end{aligned}$$

mientras

```
$$
\leqalignno{
1+2+3 &= 6, &(10) \cr
1 \times 2 \times 3 &= 6. \cr}
$$
```

genera la fórmula:

$$\begin{aligned} 1+2+3 &= 6, \\ 1 \times 2 \times 3 &= 6. \end{aligned} \tag{10}$$

Obsérvese que el segundo `&` no hace falta si la línea no tiene etiqueta.

Para colocar una fórmula centrada entre dos líneas de un despliegue, basta usar `\eqalign{...}` o `\leqalign{...}` seguido por `\eqno`. Esto es así porque estos comandos crean *bloques centrados* dentro de cajas verticales; para alinearse con otras partes de la fórmula, quedan verticalmente centradas. Por ejemplo, se obtiene:

$$(10) \iff \begin{aligned} 6 &= 1 + 2 + 3 \\ 6 &= 1 \times 2 \times 3 \end{aligned} \qquad (10^2)$$

al teclear

```
$$
(10) \iff \eqalign{ 6 &= 1 + 2 + 3 \cr
                    6 &= 1 \times 2 \times 3 \cr} \eqno(10^2)
$$
```

Obsérvese que la ⟨*etiqueta*⟩ después de `\eqno` es una subfórmula y que puede llevar exponentes, etc., al igual que la fórmula principal.

5 Macros

Un programa tan vasto y complejo como TEX se volvería inmanejable si no hubiera una forma de simplificar su uso mediante el empleo de *macros*. Un "macro" es un comando que sustituye una colección más larga de caracteres y comandos, para abreviar o sistematizar trozos de texto, y así ahorrar esfuerzo mental y tiempo de tecleo. Encontramos macros en varios niveles de complejidad.

5.1 Abreviaturas simples

Para tecleo en español, el siguiente macro puede ser útil:

```
\def\1{\'{\i}}
```

que define un nuevo macro `\1` como abreviatura del juego de 6 caracteres, `\'{\i}`, reconocido por Plain TEX. Después del renglón en el archivo fuente donde aparece esta definición, se acepta `\1` como sinónimo de `\'{\i}` y produce la í tildada.

⟦ Este macro es obsoleto en las versiones actuales de TEX, pues el carácter tecleado `\'{\i}` es reconocido y puede emplearse directamente. Es de notar, sin embargo, que es necesario establecer la equivalencia entre `\'{\i}` y el código `\'{\i}`, mediante algún archivo auxiliar, antes de emplear la `\'{\i}` tecleada en vez de `\1`; y que, debido a la incompatibilidad entre las tablas de caracteres de diferentes sistemas operativos, la versión más primitiva `\1` puede usarse en documentos que se pretenden transferir a otros sistemas. ⟧

El comando `\def` (definición) crea nuevos macros. Su sintaxis es:

`\def`⟨*comando*⟩`{`⟨*texto sustituto*⟩`}`

donde el ⟨*comando*⟩ debe ser un símbolo de control, una palabra de control, o bien un "carácter activo". La mayoría de los comandos reconocidos por Plain TEX son macros también: de unos 900 comandos reconocidos, solamente 325 comandos son "primitivos" de TEX, cuyo significado depende directamente del programa. Por ejemplo, `\def` es primitivo (por supuesto) pero `\neq`, `\>` y `~` son macros. Plain TEX contiene las siguientes definiciones:

```
\def\neq{\not =}
\def\>{\mskip\medmuskip}
\def~{\nobreak\ }
```

(La liga `~` es un "carácter activo": aunque no lleva vara, se comporta como comando y su definición puede cambiarse.)

Los macros más simples (y más usados) son las sencillas abreviaturas:

```
\def\del{\partial}
\def\la{\lambda}
\def\p{\cdot}
\def\x{\times}
\def\UCR{Universidad de Costa Rica}
\def\BibUCR{Biblioteca de la \UCR}
```

los cuales permiten teclear fórmulas y textos como:

`$a \p b \x c = a \x b \p c$`	$a \cdot b \times c = a \times b \cdot c$
`$(\la - 1)^3 = \la^3 - 3\la^2 + 3\la - 1$`	$(\lambda - 1)^3 = \lambda^3 - 3\lambda^2 + 3\lambda - 1$
`Las demoras son largas en la \BibUCR`	Las demoras ...

Obsérvese que el macro `\BibUCR` contiene otro macro como parte de su definición. Esto es factible porque los macros se "expanden" en forma automática. Más precisamente, cada `\def` agrega un comando a un "diccionario" temporal de macros. Al encontrar un comando como `\BibUCR` en el texto, el cual no es parte de Plain TEX, el programa busca este comando en el diccionario y lo reemplaza por el "texto sustituto" `Biblioteca de la \UCR`; al leer este texto, TEX encuentra el macro `\UCR`, y se repite el ciclo. La "expansión total" de `\BibUCR` es entonces la frase 'Biblioteca de la Universidad de Costa Rica'.

Las expresiones que se abrevian en macros pueden ser asaz complicadas. Ejemplos:

```
\def\mate{matem\'atica}
\def\unoaene{1,2,\ldots,n}
\def\sumkon{\sum_{k=0}^n}
\def\abcd{\pmatrix{a & b \cr c & d \cr}}
```

Obsérvese que en el último ejemplo, la plantilla `\pmatrix{a & b \cr c & d \cr}` contiene llaves. Es lógico que sea la segunda llave derecha `}` y no la primera (que sigue `&d \cr`) la que cierra el "texto sustituto". Para asegurar que así sea, TEX demanda que el texto sustituto sea "equilibrado", es decir, que cada llave `{` que abre un grupo sea seguido por una llave `}` que cierre este grupo, dentro del "texto sustituto".

La necesidad de llaves internas se ve con el siguiente ejemplo. La definición

```
\def\R{\bf R}
```

produce un efecto no deseable: cuando se usa `\R` en el archivo fuente, se efectua un cambio al estilo negrilla, porque `\R` es remplazado por `\bf R`; así, la frase

```
los n\'umeros reales \R\ forman un cuerpo ordenado
```

produciría, en impresión:

los números reales **R forman un cuerpo ordenado**

y el cambio a letra negrilla no se limita a **R**. Se puede evitar esto al encerrar `\R` en un grupo:

```
los n\'umeros reales {\R} forman un cuerpo ordenado
```

pero así se pierde toda la utilidad del macro, cuyo propósito principal es de evitar el tecleo de llaves. La definición correcta es

```
\def\R{{\bf R}}
```

pues ahora `\R` se reemplaza por `{\bf R}` y el cambio de estilo queda localizado. Las llaves externas *delimitan* el "texto sustituto" y se eliminan; las llaves internas *forman parte* del texto sustituto y se conservan en la expansión del macro.

5.2 Macros con un parámetro

La próxima etapa en la elaboración de macros es la inclusión de "parámetros" que representan argumentos. Por ejemplo, un escrito sobre álgebra lineal podría contener muchas instancias de vectores como $(x_1,\ldots,x_n)$, $(y_1,\ldots,y_n)$, $(z_1,\ldots,z_n)$, etc. Es deseable teclear `$\vector x$`, `$\vector y$`, `$\vector z$` para representar estas frases simbólicas. La idea es que `\vector` sea un macro que tome un argumento (`x`, `y` ó `z`, según el caso) y lo sustituye en lugares adecuados en el "molde" $(\square_1,\ldots,\square_n)$.

Lo que hay que teclear es:

```
\def\vector#1{(#1_1,\dots,#1_n)}
```

donde el parámetro `#1` representa el eventual argumento de `\vector`.

En esta definición, `#1` significa la "ficha" (carácter o comando) que sigue inmediatamente después de `\vector`, (habido cuenta de la eliminación de espacios en blanco). Así, `$\vector\alpha$` producirá $(\alpha_1,\ldots,\alpha_n)$.

Si la ficha que sigue `\vector` es un abre-grupo `{`, el argumento del vector es todo el grupo. Por ejemplo, `$\vector{2x}$` produce $(2x_1,\ldots,2x_n)$ porque el parámetro `#1` es sustituido por el contenido '`2x`' del grupo y se inserta '$2x$' en el molde cada vez que `#1` aparece en la plantilla del macro.

Otro ejemplo es el macro `\centerline` de Plain TEX (*TEXbook*, p. 353):

```
\def\centerline#1{\line{\hss#1\hss}}
```

Cada instancia de `\centerline{`⟨*texto*⟩`}` se expande en `\line{\hss`⟨*texto*⟩`\hss}`. A su vez, `\line` es un macro que toma un argumento y su definición es

```
\def\line{\hbox to\hsize}
```

así que `\line{\hss`⟨*texto*⟩`\hss}` se expande en `\hbox to\hsize{\hss`⟨*texto*⟩`\hss}`. Por lo tanto, `\centerline` crea una caja cuya anchura es la de la página donde aparece, y dentro de esta caja coloca su argumento con ciertos "resortes" antes y después, para centrar el texto y dejarlo proyectar en los márgenes.

Es muy aconsejable usar macros con parámetros para posponer decisiones finales acerca del formato del resultado. Por ejemplo, la definición

```
\def\reciproco#1{{1\over#1}}
```

permite obtener `$\reciproco7 - \reciproco8 = \reciproco{56}$` en la forma $\frac{1}{7}-\frac{1}{8}=\frac{1}{56}$. Si el formato de este resultado no satisface nuestro propósito, no hay que modificar el código que lo produce, sino que es cuestión de cambiar la definición del macro `\reciproco`. Por ejemplo, si usamos

```
\def\reciproco#1{1/#1},
```

el mismo código genera el resultado: $1/7-1/8=1/56$.

5.3 Macros con dos o más parámetros

El poder de los macros se pone de manifiesto cuando se emplean dos parámetros en una definición. Por ejemplo,

```
\def\fila#1#2{(#1_1,\ldots,#1_{#2})}
```

es apropiado para vectores con dimensiones variables:

`\fila xn`	$(x_1,\ldots,x_n)$
`\fila ym`	$(y_1,\ldots,y_m)$
`\fila a6`	$(a_1,\ldots,a_6)$
`\fila\alpha{p+q}`	$(\alpha_1,\ldots,\alpha_{p+q})$

La regla de sustitución es sencilla: cada vez que `\fila` aparece en el archivo fuente, se buscan las dos fichas (o grupos) inmediatamente siguientes. La primera reemplaza `#1` y la segunda reemplaza `#2`; en ambos casos, se desechan las llaves exteriores que demarcan un grupo. Por ejemplo:

En `\fila xn`,	`#1` $\longleftarrow x$,	`#2` $\longleftarrow n$;
en `\fila\alpha{p+q}`,	`#1` $\longleftarrow \alpha$,	`#2` $\longleftarrow p+q$;
en `\fila{ab}5`,	`#1` $\longleftarrow ab$,	`#2` $\longleftarrow 5$;
en `\fila{3x}{2n}`,	`#1` $\longleftarrow 3x$,	`#2` $\longleftarrow 2n$.

Figura 5.1. Sustitución de argumentos en macros

Un macro muy favorecido por los matemáticos es

```
\def\frac#1#2{{#1\over#2}}
```

que permite teclear $1 + \frac{1}{4} = \frac{5}{4}$ en la forma `$1 + \frac14 = \frac54$` en vez de la forma básica, que es más incómoda: `$1 + {1\over 4} = {5\over 4}$`, pues el macro se hace cargo de colocar las llaves obligatorias en los lugares correctos. Se abandona la sintaxis ordinaria de una fracción (númerador sobre denominador) y en cambio se obtiene la sintaxis *funcional*: una fracción es una función de su numerador y de su denominador.

El manual de LaTeX aconseja a sus usuarios teclear `\frac{3}{5}` en vez de `\frac35` para obtener $\frac{3}{5}$. En ambos casos, las sustituciones son:

`#1` $\longleftarrow 3$, `#2` $\longleftarrow 5$,

y la expansión del macro es `{3\over 5}`. Siempre se pueden rodear argumentos de un macro con llaves, aunque algún argumento conste de una sola ficha, porque las llaves exteriores se desechan antes de la sustitución de parámetros. Esto resalta los argumentos del macro al costo de teclear pares de llaves redundantes, y es una buena práctica para principiantes. Sin embargo, con un poco de experiencia se aprende a desechar las llaves redundantes.

TeX permite hasta un máximo de 9 parámetros en un macro. La única restricción que impone es que el orden natural `#1#2...#9` debe usarse después del comando por definir; dentro de la plantilla, los parámetros pueden ocurrir en cualquier orden (o no ocurrir del todo). Por ejemplo, en la teoría del momento angular, se usan los llamados "coeficientes de Clebsch–Gordan", que algunos autores escriben $C^{j_1 j_2 j}_{m_1 m_2 m}$, otros usan $\langle j_1 m_1 j_2 m_2 | j m \rangle$, algunos ponen $\langle j_1 m_1 j_2 m_2 | (j_1 j_2) j m \rangle$, y también hay autores que usan $\left\langle \begin{matrix} j_1 & j_2 \\ m_1 & m_2 \end{matrix} \middle| \begin{matrix} j \\ m \end{matrix} \right\rangle$. En cada caso, se debe teclear

```
\cleb{j_1}{j_2}{j}{m_1}{m_2}{m}
```

y definir (o redefinir) el macro `\cleb` según el formateo deseado. Las tres primeras variantes se obtienen por:

```
\def\cleb#1#2#3#4#5#6{C^{#1#2#3}_{#4#5#6}}
\def\cleb#1#2#3#4#5#6{\langle#1#4#2#5|#3#6\rangle}
\def\cleb#1#2#3#4#5#6{\langle#1#4#2#5|(#1#2)#3#6\rangle}
```

y se pueden obtener otras variantes por simples ajustes a éstas.

5.4 Macros delimitados

TeX puede delimitar el alcance de los parámetros de un macro sin tener que encerrar sus argumentos entre llaves. La sintaxis general de una definición es

`\def`⟨*comando*⟩⟨*texto paramétrico*⟩`{`⟨*texto sustituto*⟩`}`

donde ⟨*comando*⟩ y ⟨*texto sustituto*⟩ siguen las reglas ya explicadas, y ⟨*texto paramétrico*⟩ es una cadena de caracteres (posiblemente vacía) entre el ⟨*comando*⟩ y la primera llave `{` de apertura: luego, el ⟨*texto paramétrico*⟩ no puede contener llaves; además, los parámetros `#1`, `#2`, etc., deben aparecer en dicho texto solamente en orden numérico.

Los macros delimitados se usan mayormente para formatear párrafos especiales. El enunciado de un teorema en Plain TeX es formateado con '`\proclaim`':

```
\proclaim Teorema 2.4. Los n\'umeros primos $p > 2$ son impares.
```

produce:

Teorema 2.4. *Los números primos $p > 2$ son impares.*

La definición de `\proclaim` (*TEXbook*, p. 202) es esencialmente la siguiente:

```
\def\proclaim#1. #2\par{\medbreak\noindent
                        {\bf#1.\enspace}{\sl#2}\par\medbreak}
```

Los `\medbreak` producen una separación vertical extra antes y después del enunciado del "teorema". El `\noindent` suprime la sangría al inicio de este párrafo, para que el identificador esté justificado a la izquierda. El primer argumento es todo lo que antecede la primera ocurrencia de '`.␣`' (en el ejemplo, la frase '`Teorema 2.4`') y el segundo argumento es lo que sigue, hasta la próxima línea en blanco. Obsérvese que *se desechan* los separadores de argumentos '`.␣`' y '`\par`', y que el texto sustituto del macro debe restaurarlos o reemplazarlos por algo apropiado. Aquí '`.␣`' es reemplazado por '`.\enspace`', que da un espacio un poco más ancho que un espacio normal; y el '`\par`' es restaurado explícitamente. El efecto total es de crear un párrafo especial sin sangría, destacado por separación vertical del resto del texto, con un identificador en negrilla y el resto del texto en letra oblicua. Este formateo no tiene efecto alguno sobre el párrafo que sigue.

5.5 Macros condicionales

En el siguiente nivel de complejidad, se encuentran macros que pueden tomar diversas acciones que dependen de las circunstancias. Por ejemplo, muchas acciones de formateo de página (poner encabezados, cambiar los márgenes, etc.) dependen de si el número de página es par o impar. Este número es un parámetro `\pageno` de Plain TEX, pero normalmente no se sabe de antemano en cuál página quedará el resultado de lo que se teclea en el archivo fuente. Se requiere un *macro condicional* para prevenir los dos posibles casos. Se puede teclear

```
\ifodd\pageno \accionderecha \else \accionizquierda \fi
```

y se ejecutará el macro `\accionderecha` si el número `\pageno` es impar, pero `\accionizquierda` si `\pageno` es par.

El formato general de una frase condicional es:

`\if`⟨*condición*⟩ ⟨*texto-V*⟩ `\else` ⟨*texto-F*⟩ `\fi`

donde `\if`⟨*condición*⟩ empieza con un comando cuyas primeras letras son '`if`'; si la condición se cumple, se lee el '⟨*texto-V*⟩' y se ignora '`\else`⟨*texto-F*⟩'; si la condición no se cumple, se ignora '⟨*texto-V*⟩' y se lee el '⟨*texto-F*⟩'. Cualquiera de los dos 'textos' puede omitirse, en cuyo caso el macro no producirá efecto alguno en el caso correspondiente; si '⟨*texto-F*⟩' es vacío, el '`\else`' puede omitirse también.

Por ejemplo, si `\entero` es un macro cuyo valor es un número entero, podemos definir

```
\def\divisiblepordos{\entero\ \ifodd\entero no\ \fi es divisible por~2}
```

para imprimir el estado de división por 2 del número `\entero`. Se incluyen o se omiten los tres caracteres '`no␣`' en la frase, dependiendo de la paridad de `\entero`.

He aquí una lista parcial de los macros condicionales disponibles en TEX:

`\ifodd`⟨*entero*⟩	Prueba si un número entero es impar.
`\ifnum`⟨*entero*$_1$⟩ `=` ⟨*entero*$_2$⟩	Compara dos números enteros.
`\ifnum`⟨*entero*$_1$⟩ `<` ⟨*entero*$_2$⟩	Compara dos números enteros.
`\ifnum`⟨*entero*$_1$⟩ `>` ⟨*entero*$_2$⟩	Compara dos números enteros.
`\ifdim`⟨*longitud*$_1$⟩ `=` ⟨*longitud*$_2$⟩	Compara dos longitudes.
`\ifdim`⟨*longitud*$_1$⟩ `<` ⟨*longitud*$_2$⟩	Compara dos longitudes.
`\ifdim`⟨*longitud*$_1$⟩ `>` ⟨*longitud*$_2$⟩	Compara dos longitudes.
`\ifmmode`	Prueba si se está en el "modo matemático".
`\ifhmode`	Prueba si se está en el "modo horizontal".
`\ifx`⟨*ficha*$_1$⟩⟨*ficha*$_2$⟩	Prueba si dos caracteres o macros son iguales.

Figura 5.2. Las condicionales de TEX más usadas

Para obtener la letra α en texto, se puede teclear `$\alpha$`. Si ocurre muchas veces, uno puede abreviar: `\def\a{$\alpha$}`. Pero esta abreviatura no funcionará en fórmulas, porque la expansión del macro `\a` resulta en

```
$\sin\a = -\sin(\pi-\a)$   ⟼   $\sin$\alpha$ = -\sin(\pi-$\alpha$)$
```

y ahora los '`\alpha`' quedan *fuera* de modo matemático, dando lugar a varias protestas de TEX. Lo correcto es definir el macro

```
\def\a{\ifmmode \alpha \else $\alpha$\fi}
```

y de esta forma el '`\alpha`' queda siempre en modo matemático.

Hay un macro condicional `\ifcase` que hace una enumeración de casos. Su sintaxis es

```
\ifcase⟨entero⟩ ⟨texto0⟩\or ⟨texto1⟩\or ⟨texto2⟩\or ...
                \or ⟨texton⟩\else ⟨otros casos⟩\fi
```

El número ⟨*entero*⟩ debe tomar un valor (k, digamos) no negativo; se ejecuta uno de los casos $k = 0, 1, 2, \ldots, n$, siendo n el número de ocurrencias de '`\or`'. Si $k > n$, se ejecuta ⟨*otros casos*⟩.

El *TEXbook* (p. 406) explica como imprimir la fecha en forma automática (usando el reloj interno de la computadora):

```
\def\today{\ifcase\month\or
          January\or February\or March\or April\or May\or June\or
          July\or August\or September\or October\or November\or
          December\fi\ \number\day, \number\year}
```

Probamos si el número `\entero` es un primo de un dígito:

```
\ifnum\entero < 0 \number\entero\ es negativo
 \else\ifnum\entero > 9 \number\entero\ es mayor que 9
  \else\ifcase\entero \or\or\esprimo\or\esprimo\or\or
              \esprimo\or\or\esprimo\fi\fi\fi
\def\esprimo{\number\entero{} es primo}
```

Obsérvese que cada `\if...` debe terminar con su propio `\fi`; esta es una medida de seguridad para que las condicionales siempre estén correctamente encajadas.

6 El modo de funcionar de TEX

6.1 La unidad lectora de TEX

Un manuscrito o "archivo fuente" consta de líneas, pero estas líneas no tienen mucho que ver con los renglones que salen impresos. Esto es así porque los retornos (↵) son tratados como espacios de la barra de espaciamiento (␣), excepto que dos retornos seguidos (↵↵) cuentan como un fin de párrafo. Ahora exploraremos con más detalle cómo es que TEX "lee" el archivo fuente.

Un teclado envía a la computadora diversas señales eléctricas, una por cada tecla, y estas señales son los "caracteres" que la máquina recibe. Con un teclado en inglés, sin usar la tecla de "Opción", hay a lo sumo 128 caracteres posibles (incluyendo los caracteres "invisibles", como retornos ↵, tabulaciones →|, fines de página, etcétera). Estos son los 128 "caracteres ASCII".*

Es posible obtener hasta 256 caracteres, numeradas de 0 a 255, con las versiones actuales de TEX, pero los de 128 a 255 (ASCII extendido) no obedecen a un convenio universal; son distintos en

* ASCII = American Standard Code for Information Interchange.

diversos sistemas operativos. Es aconsejable usar solamente los códigos inferiores a 127 en cualquier archivo de TEX que se piensa transferir a un aparato con un convenio distinto.

TEX agrupa los caracteres en 16 "categorías", numeradas de 0 a 15. Algunos, como ⟨*nulo*⟩ y ⟨*delete*⟩ no se obtienen del teclado directamente, pero a veces se generan en procesadores de palabras como Microsoft Word, y de este modo pueden entrar al archivo fuente:

Categoría	*Símbolo*	*Significado*
0	`\`	Carácter de escape
1	`{`	Inicio de grupo
2	`}`	Fin de grupo
3	`$`	Cambio matemático
4	`&`	Tabulador
5	↵	Fin de línea
6	`#`	Parámetro
7	`^`	Exponente
8	`_`	Subíndice
9	⟨*nulo*⟩	Ignorado
10	␣	Espacio
11	⟨*letra*⟩	Letra
12	⟨*otro*⟩	Otro
13	`~`	Activo
14	`%`	Comentario
15	⟨*delete*⟩	Inválido

Figura 6.1. Las categorías de caracteres que TEX establece

Las categorías 11 y 12 contienen la mayoría de los caracteres. Una ⟨*letra*⟩ puede ser mayúscula: `A`, `B`, `C`, ..., `Z`, o minúscula: `a`, `b`, `c`, ..., `z` (pero la `Ñ` y la `ñ` no son letras en versiones de TEX anteriores a 1990). Cualquier otro carácter visible del teclado inglés tiene la categoría de ⟨*otro*⟩. Ellos son: los dígitos `1234567890`, los caracteres de puntuación `'!()-[]'";:,./?`, los símbolos aritméticos `=+<>|`, el asterisco `*`, y la arroba `@`.

Cualquier carácter visible que no aparece en estas listas (como letras acentuadas, incluyendo la `ñ`), no forma parte del juego básico de 128 caracteres ASCII. Plain TEX no les asigna categoría alguna. Para poder usarlos en el archivo fuente, se les asigna la categoría 13 (activo) de la liga `~` que los convierte en comandos y que permite asignarlos una expansión en términos de los caracteres de ASCII estándar. Vea, por ejemplo, el archivo `option_keys` que acompaña el programa *Textures*.

Nota: El carácter "invisible" ⇥(la tecla de tabulación) recibe categoría 10 cuando TEX lo lee; en otras palabras, una ⇥ *se lee como un espacio* en blanco; esto por cuanto TEX tiene su propia manera de sangrar párrafos.

6.2 Dimensiones

6.2.1. Unidades de longitud.

TEX reconoce un buen surtido de unidades de longitud, o "dimensiones", que se usan para medir arreglos tipográficos. Sabe de *pulgadas* (en el hemisferio occidental, las hojas de papel de tamaños carta y oficio tienen sus medidas y márgenes en pulgadas); de *centímetros* y *milímetros* (para hojas A4 europeos); de *puntos* y *picas*, y sus versiones europeas *didots* y *ciceros*. Además tiene un "punto grande" ("big point") para las fuentes PostScript que usan las impresoras laser, y posee su propia unidad secreta, la cual es $\frac{1}{65536}$ de un punto!

A cada unidad, TEX da un *nombre* de dos letras (una "palabra clave" que no lleva vara). He aquí la lista:

`pt`	punto	`bp`	punto grande	`dd`	didot
`pc`	pica	`cm`	centímetro	`cc`	cicero
`in`	pulgada ("inch")	`mm`	milímetro	`sp`	"scaled point"

Figura 6.2. Las dimensiones reconocidas por TEX

Además, TEX sabe que:

$$\begin{aligned} 1\,\text{pc} &= 12\,\text{pt},\\ 1\,\text{in} &= 72{,}27\,\text{pt} = 72\,\text{bp},\\ 1\,\text{in} &= 2{,}54\,\text{cm} = 25{,}4\,\text{mm},\\ 1157\,\text{dd} &= 1238\,\text{pt},\\ 1\,\text{cc} &= 12\,\text{dd}. \end{aligned}$$

Figura 6.3. Tabla de conversión para las dimensiones

El "sp" se define de modo que 1 pt = 65536 sp. Internamente, TEX convierte todas las dimensiones a "sp" para hacer sus cálculos aritméticos con números enteros; por eso sus medidas son siempre muy exactas.

⟦ Los "puntos" de los tipos de letra de las impresoras laser, hechos con PostScript, son 'bp' y no 'pt'; de ahí la distinción de las dos formas de "punto". Los tipos Computer Modern empleados por TEX usan el punto de imprenta tradicional 'pt' en sus medidas. ⟧

Para TEX, una *dimensión* es cualquier número (entero o decimal) seguido por uno de esos nueve pares de letras; así,

`3in`, `29pc`, `0pt`, `-2.5cm`, `1,3mm`, `.5pc`

son dimensiones (la "coma decimal" y el "punto decimal" pueden usarse indiferentemente para separar las partes entera y fraccional del número). Un número sin un par de letras no es interpretado como una dimensión; por ejemplo, la longitud cero no es `0`, sino `0pt` ó `0in` ó `0mm`, etc.

6.2.2. Magnificación de tipos.

Hay una manera de cambiar *todas* las dimensiones de un texto impreso de manera proporcional. Esto se hace con un *factor de magnificación*, que normalmente vale 1000. Si el archivo fuente empieza con

`\magnification = 1200`, o bien `\magnification\magstep1`,

entonces las dimensiones serán aumentadas por un 20%. (Un 'magstep' es un 20% más.) El efecto es de convertir una página de 30 pc × 40 pc, con una letra de 10 puntos, en una hoja de 36 pc × 48 pc (sólo el texto impreso, no el papel!) con una letra de 12 puntos. De igual modo, `\magnification = 800` reduce el texto en un 20%; pero es necesario disponer de los tamaños apropiados de las fuentes para obtener buenos resultados. Las versiones de TEX para sistemas Unix usualmente incluyen juegos de fuentes en el tamaño estándar y también en las magnificaciones '`\magstephalf`', '`\magstep1`', '`\magstep2`', '`\magstep3`', '`\magstep4`' y '`\magstep5`', que convierten un tipo de 10 puntos en 11 pt, 12 pt, 14 pt, 17 pt, 21 pt y 25 pt aproximadamente. Aunque estos se guardan en un formato compacto (archivos `.pk`), requieren un enorme espacio de almacenaje en disco. Las implementaciones mas modernos de TEX, como el *Textures* 2.0, usan solamente el tamaño estándar, ligado a una versión en PostScript de cada tipo.

6.2.3. Dimensiones absolutas.

Con `\magnification\magstep1` o `\magnification=1200`, se multiplican todas las dimensiones del documento por un factor de ajuste de 1.2 ($= \frac{1200}{1000}$). Así, la instrucción `\vskip 5mm` se convierte en un brinco vertical de 6 mm en impresión ($5 \times 1.2 = 6$). Dicho de otro modo, las dimensiones son medidas *relativas*. Para obtener dimensiones exactas que no dependen de la magnificación (por ejemplo, la anchura de la página), se puede decir `truein` en vez de `in`, `truecm` o `truepc` en lugar de `cm` o `pc`, etcétera. El `true` cancela el ajuste de la magnificación por una sola vez; así, `\hsize=6.5truein` significa que el texto tendrá una anchura de 6.5 in, independientemente de la magnificación de los tipos de letra.

6.2.4. Medidas que dependen del tipo.

Hay otras dos unidades de longitud que TEX reconoce: un `em` es la anchura de un cuadratín `\quad` (o de la barra larga — si ésta forma parte de la fuente en uso), y un `ex` es la altura de una letra x. Estas unidades dependen de la fuente en uso. Por ejemplo, 1 em = 10 pt en la fuente romana de 10 pt de la familia Computer Modern (`cmr10`), pero 1 em = 11.5 pt en la fuente negrilla de 10 pt (`cmbx10`). Luego, si se desea una sangría de 2 cuadratines, se puede poner `\parindent=2em` al inicio del archivo; pero en ese caso la sangría de párrafos en letra negrilla será mayor (de 23 pt en lugar de 20 pt). El formato de PlainTEX establece `\parindent=20pt`.

6.3 Cajas

Para TEX, una página de texto no es más que *un armazón de cajas*. La página es una caja rectangular; contiene un juego de tres cajas adentro: la cabeza, el cuerpo, y el pie, ordenados verticalmente de arriba para abajo. El cuerpo de la página se subdivide a su vez en una lista de párrafos, cada párrafo es una lista vertical de cajas que son los renglones, y cada renglón es una lista horizontal de cajas que son los caracteres. En estas últimas cajas, hay que colocar "tinta" para pintar el carácter en la hoja impresa; pero eso es un trabajo aparte que le toca a la impresora o al programa de prevista en pantalla. Lo que hace TEX es *decidir dónde colocar las letras* en la página.

Cada caja (sea página, fórmula, renglón o carácter) es un *rectángulo* que tiene determinadas *altura, hondura* y *anchura*; éstas longitudes se miden a partir de un *punto de referencia* en el margen izquierdo:

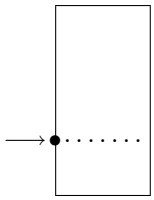

Figura 6.4. Línea de base y punto de referencia de una caja

La línea horizontal que pasa por el punto de referencia se llama la *línea de base* de la caja. Algunas de estas dimensiones pueden ser cero: por ejemplo, los siete caracteres de `palabra` constan de uno (`p`) que tiene una hondura no cero y otros seis que tienen hondura cero. Puesto que se alinean por sus líneas de base, se imprimen correctamente.

Como la puesta de tinta en las cajas es una operación independiente del montaje del juego de cajas, se permite que el "contenido" de una caja no necesariamente quede dentro de sus fronteras. Aunque esto es un tanto extraño, resulta ser muy útil. (Por ejemplo, de este modo se pueden colocar notas en los márgenes de un texto, fuera del borde del "cuerpo de página".)

Una caja que está completamente llena de tinta se llama una *regla*. Una regla de poca altura o hondura:

---

es una regla horizontal: `\hrule`; una regla de poca anchura | es una regla vertical: `\vrule`. Un rectángulo negro ▮ puede ser un `\hrule` o un `\vrule` según las circunstancias. Hay que usar un `\hrule` entre párrafos y un `\vrule` dentro de un párrafo.

Las cajas se alinean horizontal o verticalmente. Una lista horizontal de cajas (los caracteres en un renglón, por ejemplo) se alinean y se empacan en una caja horizontal: `\hbox`. Una lista vertical de cajas se alinean y se empacan en una caja vertical: `\vbox`. Si uno quiere hacer esta operación explícitamente, para obtener un efecto especial, uno puede formar sus propios `\hbox{...}` y `\vbox{...}`. Por ejemplo:

```
\vbox{\hbox{Dos renglones}        Dos renglones
      \hbox{de texto.}}           de texto.
```

6.4 Goma

Las cajas se pegan con *goma*. Mejor dicho, la "goma" es el espacio que *separa* cajas en vez de unirlas; se llama así porque este es un espacio que puede *estirarse o contraerse* para obtener la separación deseada entre las cajas.

6.4.1. Flexibilidad de la goma.

Un renglón de texto contiene aproximadamente unas 70 cajas: las *letras* que lo componen. Estas se agrupan en palabras: cada palabra consta de varias cajas de letra sin separación. Entre las palabras está la goma, la cual se estira o se contrae para justificar el renglón en los márgenes. También hay goma entre los renglones: esta goma flexible se ajusta de tal modo que las líneas de base de los renglones de un párrafo se separen siempre por el mismo monto. En un texto de letra de 10 pt, las líneas de base se separan normalmente por 12 pt; en la jerga de imprenta, tenemos "letra de 10 pt sobre una base de 12 pt".

Cada gota de goma tiene *tres medidas*: su *tamaño natural*, su *estirabilidad*, y su *contractibilidad*. Un espacio en blanco (producido por ␣ o \␣ o ↵ o ⊣) genera una gota de goma y no una caja, y su tamaño natural produciría renglones no justificados. No se contrae más allá de su "contractibilidad": si el renglón no puede justificarse aun con la máxima contracción, se proyecta al margen y da un `Overfull \hbox`. Si, por otro lado, el renglón no puede justificarse aun con el máximo estirón de cada uno sus gotas de goma, se le permite estirar un poco más, pero TEX protesta acerca de un `Underfull \hbox`. Si una página se corta sin tener suficiente espacio en blanco entre sus párrafos, la "goma vertical" colocada allí se estira, pero TEX emite una protesta sobre un `Underfull \vbox while output is active`.

Una gota de goma se obtiene mediante

`\hskip`⟨*dimen*$_1$⟩ `plus`⟨*dimen*$_2$⟩ `minus`⟨*dimen*$_3$⟩, o bien
`\vskip`⟨*dimen*$_1$⟩ `plus`⟨*dimen*$_2$⟩ `minus`⟨*dimen*$_3$⟩

donde ⟨*dimen*$_1$⟩ es la longitud "natural" de la goma, ⟨*dimen*$_2$⟩ es el máximo estirón normalmente permitido, y ⟨*dimen*$_3$⟩ es la máxima contracción permitida. (Obsérvese que las palabras claves `plus` y `minus` no llevan vara adelante.) Por ejemplo, un `\medskip` se define en Plain TEX como:

```
\def\medskip{\vskip6pt plus2pt minus2pt}
```

y normalmente da un brinco vertical de 6 pt, pero puede variar entre 4 pt y 8 pt, dependiendo de las exigencias de la confección de página. Fíjese que las ⟨*dimen*⟩ deben llevar la unidad de medición siempre.

Un `\hskip` produce un espacio horizontal y un `\vskip` resulta en un brinco vertical. Por ejemplo, un espacio de medio cuadratín (5 pt para la fuente `cmr10`) se obtiene con `\enskip`, que es una abreviatura para `\hskip.5em\relax`. En un `\enskip`, la longitud del espacio es fija; el papel del `\relax` (que no hace nada) es de prevenir la posibilidad de que el texto que sigue empieza con las letras `plus...` o `minus...`

Si varias gotas de goma se estiran o se contraen simultáneamente, el monto del cambio de cada gota es proporcional a su estirabilidad o contractibilidad respectivamente. Hecho este ajuste, se empaca el resultado en una caja más grande que luego queda fija: se dice que se está "secando la goma". (Así es cómo se forman las cajas de los renglones.)

6.4.2. Brincos verticales entre párrafos.

Los espacios entre renglones normalmente están determinados por el estilo general del documento, y no es necesario modificarlos. Si, por ejemplo, el texto tiene letras de 10 pt, se establece una "base" de 12 pt; si la altura de un renglón más la hondura del renglón anterior suman menos de 12 pt, se coloca goma entre las cajas de renglón para que la separación vertical entre las líneas de base sea exactamente 12 pt. Si se desea más espacio entre párrafos (por ejemplo, para destacar un cambio de tema), se puede agregar un `\smallskip` *entre párrafos*: un `\smallskip` es una gota de goma que mide 3 pt (en su tamaño natural). Más grande es un `\medskip` (un brinco mediano), que es de 6 pt, más o menos; y entre secciones de un documento conviene poner un `\bigskip`, que da aproximadamente 12 pt de espacio vertical.

6.4.3. Resortes.

Hay una especie de goma cuya estirabilidad es "infinita". Como varias gotas de goma en una lista se estiran *proporcionalmente*, si una de ellas pudiera estirarse indefinidamente, entonces las

demás no se estirarían. A una unidad de esta goma le daremos el nombre de **resorte**. Hay resortes horizontales: `\hfil`, y resortes verticales: `\vfil`. Como es de esperar, un `\hfil` se usa dentro de un párrafo (o en un `\hbox`) y un `\vfil` se usa entre párrafos de texto (o en un `\vbox`).

Si hay un resorte en el extremo izquierdo de un renglón, al estirarse el texto se corre hacia la derecha, "justificando" el renglón a la derecha. Si se ponen resortes en ambos extremos de un renglón, se estiran por igual monto y el texto *queda centrado*. Así es cómo funciona `\centerline`.

Otros resortes más poderosos son `\hfill` y `\vfill`. Su estirabilidad es mucho mayor que la de `\hfill` y `\vfill`; cuando se suman un `\hfill` y un `\hfil`, el efecto del segundo queda anulado.

Al fondo de una página, `\vfill` proporciona un relleno vertical con espacio en blanco. La despedida `\bye` coloca este tipo de goma automáticamente en la última página de un documento.

Además, los comandos `\hss` y `\vss` producen goma que puede estirarse y también contraerse indefinidamente. Por ejemplo, el comando `\line{`⟨*texto*⟩`}` pone un texto en una caja de la anchura de `\hsize`. Entonces `\line{\hss`⟨*texto*⟩`\hss}` produce una caja centrada, que puede proyectar en los márgenes sin causar una protesta sobre un `Overfull \hbox`: es así como se define `\centerline`.

6.4.4. Goma después de puntuación.

En texto ordinario, se coloca el mismo monto de goma entre las palabras de una oración. Pero después de puntuación, se coloca goma de mayor tamaño natural, con más estirabilidad y menos contractibilidad. En consecuencia, un espacio después de un punto o una coma es levemente más larga que un espacio después de una palabra que termina sin puntuación. Luego es innecesario (aunque permisible) teclear dos espacios después de puntos, comas, dos puntos, punto y coma, etcétera. TEX, con sus gotas de goma, hará lo que es apropiado en cada caso.

Sin embargo, hay ciertas excepciones a las reglas automáticas de TEX que requieren una intervención explícita del tecladista. A veces uno quiere obtener el espacio usual entre dos palabras, aunque la primera lleve puntuación. He aquí una pequeña lista de casos especiales:

- ◇ *Puntos suspensivos.* Tres puntos tecleados juntos `...` producen ... con separación insuficiente, y tres puntos separados daría un resultado peor: . . . Luego Plain TEX tiene el comando `\dots` cuyo efecto `\dots` tiene la medida correcta de una elipsis textual.
- ◇ *Abreviaturas.* Las abreviaturas como Sr., Srta., Dr., Prof., i. e., p. ej., etc., no terminan oraciones, aunque cada una es una palabra seguida por un punto. En estos casos, se puede usar la *liga* `~`, que produce un espacio ordinario: `Sr.~Juez`, `Prof.~Girafales`, `p.~ej.~veamos`, `etc.~etc.` Alternativamente, se puede emplear el espacio explícito `\␣` que produce un espacio ordinario, al igual que la liga, pero permite que el renglón se rompa en ese lugar. Esto es útil, por ejemplo, al citar revistas: `Rev.\ Fil.\ Univ.\ Costa Rica`.
- ◇ *Espaciamiento uniforme.* En bibliografías, donde ocurren muchas citas con abreviaturas, es apropiado que *todos* los espacios sean de la misma anchura, independientemente de la puntuación. (Esto elimina la necesidad de estar tecleando tantos `\␣`.) Se obtiene este efecto al teclear `\frenchspacing` antes de la bibliografía. Se puede regresar al método de espaciamiento usual al teclear `\nonfrenchspacing`.
- ◇ *Después de una letra mayúscula.* Un punto que sigue una letra mayúscula es seguido por un espacio normal (sin necesidad de teclear `\␣`). `Donald~E. Knuth`, el creador de TEX, así lo estableció para poder teclear iniciales de nombres (o de segundos apellidos) en forma sencilla.

6.4.5. Interlineación.

¿Cómo asegura TEX que los renglones de un párrafo estén espaciados uniformemente? Cada renglón es una caja con una línea de base, una altura y una hondura. Hay un parámetro de referencia, que es una gota de goma llamada `\baselineskip`, que en Plain TEX es de 12 pt (sin más ni menos) pero que puede modificarse en otros estilos. Entonces, si la altura del renglón actual, más la hondura del renglón anterior, resulta menor que `\baselineskip`, TEX inserta una gota de goma vertical entre estos dos renglones para ajustar la distancia entre sus líneas de base al valor de `\baselineskip`. Esta separación canónica entre líneas de base se llama *interlineación* ("leading", en inglés); anteriormente se colocaban pequeñas regletas de plomo en la plancha de impresión. Si la altura más la hondura mide más que `\baselineskip`, no hay que insertar nada, y el segundo

renglón queda empujado hacia abajo: tal es el caso si, por ejemplo, uno de los dos renglones contiene una fracción grande en el estilo `\displaystyle`.

⟦ Puede darse el caso, bajo este régimen, de dos renglones muy cercanos (separados por un décimo de milímetro, digamos). Para prevenir ese caso, TEX proporciona otros dos parámetros, `\lineskiplimit` y `\lineskip`. Si la separación calculada por el procedimiento anterior es menor que `\lineskiplimit`, se desecha el cálculo y se coloca un brinco de `\lineskip` entre los dos renglones. Plain TEX establece `\lineskip = 1pt`. ⟧

Es posible quitar la goma vertical extra de entre dos cajas en una lista vertical, al decir `\nointerlineskip` entre estas dos cajas. Este recurso es muy valioso en la confección de tablas regladas.

6.5 Modos

Internamente, TEX puede trabajar en 6 *modos* diferentes, dependiendo de su función. La lista de estos modos es:

1a) *Modo Vertical*: cuando arma la lista vertical de cajas que componen una página.
1b) *Modo Vertical Interno*: al armar una lista vertical dentro de algún `\vbox`.
2a) *Modo Horizontal*: cuando arma la lista de caracteres y espacios que constituyen un párrafo.
2b) *Modo Horizontal Restringido*: al armar una lista horizontal dentro de un `\hbox`.
3a) *Modo Matemático*: cuando arma una fórmula en texto.
3b) *Modo Matemático de Despliegue*: cuando arma una fórmula que se desplegará en un renglón aparte.

Figura 6.5. Los modos de TEX

Al inicio de un trabajo, TEX está en el "modo vertical", listo para construir páginas. En este modo, TEX acepta comandos para brincos verticales como `\vskip`, `\smallskip`, `\bigskip`, y reglas horizontales como `\hrule`. Cuando está en su modo vertical, TEX ignora las líneas en blanco (o comandos `\par`). Por lo tanto, dos o tres líneas en blanco valen igual que una línea, para efectos de terminar un párrafo de texto.

Al encontrar texto (una letra, por ejemplo), TEX cambia a su "modo horizontal" y empieza a acumular el contenido de un párrafo. Al encontrar ↵↵ o el comando explícito `\par`,* TEX termina el párrafo, lo arma rápidamente en renglones (justificados) y regresa a su modo vertical para seguir confeccionando la página. (Si un comando propio del modo vertical, como un `\smallskip` por ejemplo, ocurre dentro del texto de un párrafo, se termina el párrafo automáticamente en ese lugar.)

Cuando TEX encuentra un signo de dólar `$`, hace un cambio a su "modo matemático" para montar la fórmula que debe seguir este signo, y sigue construyendo esta fórmula hasta encontrar otro `$`, en cuyo caso vuelve al modo horizontal. Así, el carácter `$` funciona como un *conmutador* (o "switch") que *enciende y apaga* el modo matemático.

Si dentro de un párrafo TEX ve dos signos de dólar seguidos: `$$`, el párrafo se interrumpe y se lee una fórmula que debe terminar con otro `$$`; TEX monta esta fórmula y *la coloca en un renglón aparte en forma centrada*, y luego regresa al párrafo interrumpido.

7 Formatos de texto

7.1 Párrafos

TEX inicia sus trabajos en el modo vertical, y permanece allí mientras coloca reglas, saltos verticales y crenajes verticales en la página. Al encontrar un carácter que puede imprimir, pasa a "modo horizontal" y empieza un párrafo. Más precisamente, se *inicia* un párrafo al encontrar cualquiera de estos objetos en modo vertical:

◇ un carácter visible;
◇ un signo de dólar simple `$` que empieza una fórmula;

* La unidad lectora de TEX convierte ↵↵ en `\par`; internamente, todos los párrafos de texto terminan con `\par`.

- ◇ uno de los macros `\indent`, `\noindent`, `\leavevmode`;
- ◇ cualquier comando que requiere modo horizontal para su acción apropiada: `\hskip`, `\vrule`, `\quad`, etc.

Se *termina* un párrafo al encontrar cualquiera de estos comandos:

- ◇ dos retornos de carro ↵↵ consecutivos;
- ◇ el comando explícito `\par`;
- ◇ un salto vertical: `\vskip`, `\smallskip`, `\medskip`, `\bigskip`;
- ◇ cualquier otro comando que requiere modo vertical para su acción apropiada: `\bigbreak`, `\hrule`, `\vfill`, etc.

Se *interrumpe* un párrafo al encontrar un dólar doble `$$` en su contenido; lo que sigue de ahí al siguiente `$$` se coloca, como fórmula centrada, en uno o más renglones aparte. Después del segundo `$$`, se *reinicia* el párrafo interrumpido en un nuevo renglón sin sangría. Si el párrafo termine antes de que aparezca el segundo `$$` (si, por ejemplo, la fórmula tuviera una línea en blanco en su interior), TEX protestará por manejo indebido.

7.2 Sangría

El primer renglón de un párrafo siempre lleva sangría, la cual TEX coloca en forma automática. Lo que hace TEX es colocar al margen izquierdo del primer renglón una caja vacía de anchura `\parindent`. Este parámetro es definido por el formato: Plain TEX usa `\parindent = 20pt`, LATEX usa `\parindent = 1.5em`, $\mathcal{AMS}$-TEX usa `\parindent = 1em` (un artículo de revista lleva menos sangría que un libro de texto). Para suprimir la sangría del todo, coloque `\parindent = 0pt` al inicio de su archivo.

La sangría se suprime por una sola vez con `\noindent`. Se usa `\noindent` al inicio de un capítulo o sección en algunos formatos.

La sangría puede ser *negativa*, en cuyo caso el primer renglón del párrafo proyectará en el margen izquierdo. Basta asignar `\parindent = -2em` antes del párrafo. (No olvide reasignar a `\parindent` su valor original después del párrafo o párrafos que deben llevar sangría negativa, o encerrar la asignación y el párrafo entre llaves.)

7.3 Los márgenes

Plain TEX construye párrafos de anchura `\hsize` en una hoja tamaño carta, con márgenes de 1 pulgada a cada lado. Más precisamente, la impresión se inicia 1 pulgada a la derecha del borde izquierdo del papel. Luego Plain TEX coloca `\hsize = 6.5in`, para que el margen derecho quede una pulgada a la izquierda del borde derecho de la hoja. En esta situación "default", los parámetros `\leftskip` y `\rightskip` valen 0 pt. Los márgenes pueden ser *desplazados* por asignación de nuevos valores a `\leftskip` y `\rightskip`.

Por ejemplo, para colocar una cita textual en la página, se puede hacer lo siguiente:

```
{\smallskip
 \leftskip = \parindent  \rightskip = \parindent
 ⟨texto de la cita⟩
 \smallskip}
```

y los márgenes se contraerán por el monto de una sangría en ambos lados. La separación vertical entre la cita y los párrafos ambientes es el resultado de los `\smallskip`. Obsérvese que el segundo `\smallskip` termina el párrafo; esto es necesario porque los márgenes no se calculan hasta que TEX haya leído el párrafo entero. Las llaves localizan el efecto de modificar los márgenes. El párrafo que sigue usará los márgenes anteriores.

Para mayor comodidad, Plain TEX proporciona el macro `\narrower` que tiene el mismo efecto que `\leftskip=\parindent \rightskip=\parindent`. El macro `\narrower` es acumulativo: al teclear `\narrower\narrower`, los dos márgenes se contraen por un monto de `2\parindent` cada uno.

7.4 Sangría colgante

Algunos párrafos llevan sangría, no en el primer renglón sino en todos los renglones *subsiguientes*: esto se llama "sangría colgante". Es particularmente útil para formatear *listas* de varios párrafos que deben llevar una señal "flotando" a la izquierda.

Para listas, tenemos 2 macros, `\item` e `\itemitem`. Ambos toman un argumento que se coloca a la izquierda del primer renglón, en la caja de sangría. Cada `\item` o `\itemitem` empieza un párrafo nuevo, con sangría colgante de `\parindent` o `2\parindent` respectivamente; este efecto se apaga al final del párrafo. Ejemplo:

- ◇ Las letras A, B, C se usan comúnmente para denotar los *vértices* de un triángulo. Otras letras del abecedario denotan otros puntos especiales del triángulo $\triangle ABC$.
- ◇ D, E, F denotan los pies de las alturas.
- ◇ K, L, M denotan los puntos medios de los lados.
- ◇ X, Y, Z denotan los puntos de contacto del incírculo con los lados.
- ◇ Las letras G, H, I, N, O, S se reservan para otros puntos especiales:
  - $G = AK \cap BL \cap CM$ es el centroide.
  - $H = AD \cap BE \cap CF$ es el ortocentro.
  - O es el circuncentro.
  - I es el incentro.
  - N es el centro del círculo que pasa por K, L, M, D, E, F.
  - $S = AX \cap BY \cap CZ$ es el punto de Gergonne.

Figura 7.1. Un ejemplo del uso de `\item` *e* `\itemitem`

Veamos dos renglones del archivo fuente que produjo esta lista:

```
\item{$\diamond$} $D$, $E$, $F$ denotan los pies de las alturas.
\itemitem{\bf--}  $H = AD \cap BE \cap CF$ es el ortocentro.
```

Se obtienen listas enumeradas con `\item{1.}`, `\item{2.}`, etc., y sublistas con `\itemitem{a)}`, `\itemitem{b)}`, etc. El argumento de `\item` o `\itemitem` es lo que flota a la izquierda del renglón inicial.

La sangría colgante se controla con dos parámetros, `\hangindent` (el monto de la sangría) y `\hangafter`⟨*entero*⟩, donde ⟨*entero*⟩ es el número de renglones normales que se colocan *antes* de sangrar. En un `\item`, se usa `\hangindent = \parindent` y `\hangafter = 1`. (La sangría en el primer renglón es la de siempre.)

7.5 Justificación de renglones

TEX hace grandes esfuerzos para "justificar" los renglones de un párrafo de la mejor manera posible, empleando guiones si fuera necesario. Pero en algunas ocasiones excepcionales es necesario quitar la justificación y levantar uno o más párrafos "justificados a la izquierda sólamente", o como dicen los sajones, "rasgados a la derecha". Plain TEX tiene un comando `\raggedright` que proporciona este efecto.

Para justificar uno o dos renglones solamente, tenemos `\centerline`, `\leftline` y `\rightline`: son macros que toman como argumento un solo renglón de texto y lo colocan centrado, junto al margen izquierdo o junto al margen derecho, respectivamente. El macro `\centerline` es muy útil para títulos y leyendas en tablas y figuras; `\rightline` se emplea para colocar una fecha en una carta (con `\rightline{\today}`, por ejemplo).

Si se requiere un párrafo de renglones cortos, para un poema, digamos, `\raggedright` no es suficiente. Hay que habilitar el retorno de carro ↵ para que sea un verdadero fin-de-renglón. Plain TEX proporciona `\obeylines` para estos casos: mientras esté en vigor, un renglón del archivo

fuente producirá un renglón impreso. Ejemplo: †

```
{\obeylines\smallskip
Recuerde el alma dormida,
abive el seso y despierte
\quad contemplando
c\'omo se passa la vida,
c\'omo se viene la muerte
\quad tan callando;
cu\'an presto se va el plazer,
c\'omo despu\'es de acordado
\quad da dolor,
c\'omo a nuestro parescer,
cualquiera tiempo passado
\quad fu\'e mejor.
\smallskip}
```

Recuerde el alma dormida,
abive el seso y despierte
contemplando
cómo se passa la vida,
cómo se viene la muerte
tan callando;
cuán presto se va el plazer,
cómo después de acordado
da dolor,
cómo a nuestro parescer,
cualquiera tiempo passado
fué mejor.

Figura 7.2. Cómo levantar "Coplas por la muerte de su padre"

7.6 Cortes de renglón

Un procesador de palabras corriente confecciona sus párrafos renglón por renglón: construye el primer renglón, luego el segundo, después el tercero, etcétera. Pero TEX *lee todo el párrafo primero* antes de armarlo: eso permite buscar los mejores lugares para dividir los renglones, probando varias posibilidades para maximizar la uniformidad y reducir el número de guiones de división silábica.

Todo eso se hace por un complejo algoritmo numérico; sin embargo, ningún proceso automático es perfecto, y a veces hay que avisar explícitamente a TEX que es mejor dividir el renglón en otro lugar. Los controles que el tecladista tiene para indicar dónde puede cortarse y dónde no debe cortarse un renglón son tres: *cortes obligados, guiones opcionales* y *ligas*.

7.6.1. Cortes obligados.

Para cortar una línea en un determinado lugar, teclee `\break`. Ahora, esto estirará el renglón que termina (debido a la flexibilidad de la goma entre palabras) y puede dar lugar a un `Underfull \hbox`. La otra alternativa es justificar el renglón cortado a la izquierda, con `\hfil\break`.

7.6.2. Guiones opcionales.

A veces TEX elige sus guiones erróneamente (su "diccionario" de guiones es más compacto que los de otros programas de edición, pero no es exhaustivo). Por ejemplo, Plain TEX cree que la palabra 'general' se divide en 'gen-eral', y no en 'ge-ne-ral', que es lo correcto en español. Si se ve por la ventana 'typeset' que una división incorrecta de palabra ha ocurrido, se puede volver al archivo fuente e insertar *guiones opcionales* en lugares apropiados. Si la palabra `general` en ese archivo se modifica a `ge\-ne\-ral`, en el próximo levantamiento se dividirá como 'ge-neral' o bien 'gene-ral', dependiendo de cuál de estas dos opciones conviene más para el párrafo total; el comando `\-` coloca un "guión opcional", que: (a) se ignora cuando no se convierte en guión, y (b) prohibe colocar un guión en la misma palabra excepto en los lugares señalados por guiones opcionales.

Si una palabra (como 'cualquier' o 'cerrado') se divide mal muchas veces en el levantamiento, hay otra alternativa. Coloque el comando `\hyphenation{`⟨*palabras*⟩`}` al inicio del archivo, donde ⟨*palabras*⟩ es una lista de palabras con guiones insertadas en los lugares apropiados. Por ejemplo,

```
\hyphenation{ce-rra-do cual-quier ge-ne-ral}
```

establece la división silábica de las palabras 'cerrado', 'cualquier', 'general' de una vez por todas. (En estas ⟨*palabras*⟩, no debe haber acentos, ya que `\hyphenation` no los acepta.)

† *Coplas por la muerte de su padre*: Jorge Manrique, 1440?–1479.

7.6.3. Ligas.

He aquí la lista de lugares donde el *TEXbook* recomienda colocar *ligas*:

◇ Para referirse a partes nombradas de un documento:

```
Parte~2,   Secci\'on~2.4,   Figura~6,   Ap\'endice~B.
```

◇ Entre nombres y apellidos:

```
Juan~C. del~Valle,   Pedro P\'erez~P.,   Juan Pablo~II.
```

◇ Entre símbolos matemáticos en aposición con sustantivos:

```
la variable~$x$,   una funci\'on~$f$.
```

◇ Entre símbolos en serie:

```
1,~2 o~3,   $a$,~$b$ y~$c$.
```

◇ Entre preposiciones y símbolos:

```
de~0 a~9,   restar 2 de~$x$.
```

◇ En frases matemáticas:

```
menor que~4,   2~veces el producto,   para $p$~primo.
```

◇ Cuando se enumeran casos en texto:

```
(2)~En segundo lugar, la idea es (a)~falaz; y~(b)~in\'util.
```

7.6.4. Guiones escondidos en cajas.

Para prohibir un corte de renglón *en un guión*, hay que "esconder" el guión, metiéndolo dentro de una caja. Un `\hbox` es una unidad textual que no puede romperse. Si, por ejemplo, se corta un renglón en la barra corta de la frase `p\'aginas 230--246`, se puede modificar el texto a `p\'aginas \hbox{230--246}`, y se imprime '230–246' como una unidad indescomponible.

7.6.5. Penalties.

La decisión de cortar o no cortar un renglón (en modo horizontal) es el resultado de un cálculo numérico interno, usando diversos parámetros que miden la "fealdad" de cada renglón. Se puede hacer un ajuste explícito a este cálculo con la inserción de un `\penalty`⟨*entero*⟩ en el archivo fuente: un "penalty" positivo inhibe un corte de renglón mientras un "penalty" negativo promueve un corte (al bajar su "costo" en el cómputo interno). Un "penalty" de 10000 prohibe un corte totalmente (`\nobreak` es un sinónimo de `\penalty10000`) y un "penalty" de −10000 genera un corte obligatorio (`\break` es igual a `\penalty-10000`).

⟦Los penalties insertados entre párrafos modifican los cortes entre dos páginas de texto. Si se dice `\par\penalty-50\smallskip`, se obtiene un brinco vertical `\smallskip` junto con una pequeña recomendación de cortar la página allí; el macro `\smallbreak` tiene este efecto. Se puede decir `\smallbreak`, `\medbreak` o `\bigbreak` para obtener el brinco correspondiente, con "penalties" de −50, −100 y −200 respectivamente. También hay un macro `\goodbreak`, que es un sinónimo de `\par\penalty-500`: esto sugiere la conveniencia de un corte de página sin agregar goma. Si no cae cerca del fin de la página, el `\goodbreak` no tiene efecto.⟧

7.7 Notas al pie de página

Las notas al pie de página se colocan con el comando `\footnote` (por supuesto).* Aquí se tecleó:

```
(por supuesto).\footnote*{Esta es una nota al pie.}
```

El comando `\footnote` debe seguirse por una seña de referencia, como `*`, `{**}`, `\dag`, `\ddag`, que también podría ser un número elevado, como `{$^1$}`, `{$^2$}`, etcétera; y luego por el texto de la nota, entre llaves. La seña de referencia** se pone en el párrafo y también en la nota al pie; aquí se tecleó:

```
La se\~na de referencia%
%
\footnote{**}{Como este par de asteriscos.}
%
se pone en ...
```

para producir la segunda nota abajo.[3] (Obsérvese el papel de los `%` que permiten destacar el comando `\footnote` en un renglón aparte del archivo fuente, para verlo más fácilmente a la hora de revisar este archivo para hacer correcciones.)

7.8 Inserciones de bloques flotantes

Las notas al pie de página, las tablas y las figuras son tratados por TEX como *inserciones*: antes de colocarlos, hay que asegurar que quede campo en la página. Las inserciones son cosas "flotantes": las notas se colocan al pie y las tablas y figuras al inicio o en medio de la página.

Si se desea colocar un gráfico o ilustración (hecho aparte) que tiene una altura de 5 cm (digamos) al inicio de una página, se puede reservar el sitio con el siguiente código:

```
\topinsert
\vskip 5cm \smallskip
\centerline{\bf Figura 8}
\endinsert
```

y se abrirá un espacio en blanco de 5 cm al inicio de la página, con la leyenda **Figura 8** centrado por debajo. Si no hubiere campo en una página para una inserción de este tipo, TEX lo colocará en la página siguiente (al inicio). Todo lo que hay entre `\topinsert` y `\endinsert` es un bloque (en modo vertical) que se colocará en un solo lugar. Después del bloque TEX inserta un `\bigskip` automáticamente para separar este bloque del resto de la página.

Para colocar una inserción en medio de una página, se teclea

```
\midinsert
⟨inserción⟩
\endinsert
```

en el lugar donde se la quiere colocar: será insertada allí si queda sitio; y si no, será puesto al inicio de la página actual (o si eso fuera también imposible, al inicio de la página siguiente). Para una inserción que llena una página entera, diga `\pageinsert` ⟨*inserción*⟩ `\endinsert`: por supuesto, esta inserción se colocará en la página *siguiente* al lugar en donde aparece la instrucción.

---

* Esta es una nota al pie.

** Como este par de asteriscos.

[3] Esta nota se produjo con `... abajo.\footnote{$^3$}{Esta nota ...`

8 Tabulación y Tablas

8.1 Tabulación

Normalmente se usa la tecla ⇥de tabulación, en procesadores de palabras, para sangrar y para efectuar ciertas separaciones visuales en pantalla. Pero TEX coloca la sangría al inicio de cada párrafo en forma automática y la tabulación se reserva para textos que deben alinearse.

Cada renglón con tabulación debe teclearse en la forma

`\+` ⟨*texto*$_1$⟩ `&` ⟨*texto*$_2$⟩ `&` ⟨*texto*$_3$⟩ `& ... &` ⟨*texto*$_n$⟩`\cr`

donde el ⟨*texto*$_1$⟩ comienza al margen izquierdo, el ⟨*texto*$_2$⟩ empieza en la segunda columna, etcétera. El renglón debe empezar con `\+` y debe terminar con `\cr`; cada `&` funciona como un tabulador. Las diversas columnas de texto forman "grupos" aunque no llevan llaves explícitamente: un cambio de tipografía en una columna no afecta a la columna siguiente.

Lo que queda por ver es cómo se establecen las posiciones de los tabuladores. Hay tres maneras de hacerlo. La primera, que es la más sencilla, es que la primera instancia de `\+...\cr` establezca las posiciones de los tabuladores, de acuerdo con las anchuras naturales de sus columnas. Las instancias subsiguientes de `\+` usarán las columnas ya establecidas. Ejemplo:

```
\+ \kern 3cm & \kern 2cm & \kern 2cm &\cr
\+ \sl Unidad SI &\it S\'{\i}mbolo & \it Dimensi\'on & \it Nombre\cr
\smallskip
\+ Metro     & m   & $L$      & longitud \cr
\+ Kilogramo & kg  & $M$      & masa \cr
\+ Segundo   & s   & $T$      & tiempo \cr
\+ Amp\‘ere  & A   & $I$      & corriente el\’ectrica \cr
\+ Kelvin    & K   & $\Theta$ & temperatura \cr
\+ Mol       & mol & $N$      & cantidad de sustancia \cr
\+ Candela   & cd  & $J$      & intensidad luminosa \cr
```

Este código produce:

Unidad SI	*Símbolo*	*Dimensión*	*Nombre*
Metro	m	L	longitud
Kilogramo	kg	M	masa
Segundo	s	T	tiempo
Ampère	A	I	corriente eléctrica
Kelvin	K	Θ	temperatura
Mol	mol	N	cantidad de sustancia
Candela	cd	J	intensidad luminosa

Figura 8.1. Ejemplo de tabulación con `\+ ... \cr`

El primer renglón contiene crenajes solamente, para establecer las posiciones de los tabuladores. Obsérvese el uso de `\smallskip` para separar renglones. Cada renglón entre un `\+` y el `\cr` que sigue es un párrafo completo (sin sangría) y se puede colocar goma vertical entre estos párrafos. El `\sl` que pone la frase 'Unidad SI' en estilo oblicuo no afecta el 'Metro' de la fila siguiente, que aparece en estilo romano; igualmente, el `\it` que convierte 'Símbolo' en estilo itálico no afecta 'Dimensión' de la próxima columna y hay que pedir `\it` de nuevo. También, se ignoran los espacios en blanco después de `\+` y de cada `&`, para facilitar la identificación de estos símbolos en el archivo fuente.

8.2 Colocación de tabuladores

8.2.1. Tabulación centrada.

Para *centrar* una tabulación, se puede usar un despliegue matemático, pues el contenido de un despliegue queda centrado y con cierta separación vertical arriba y abajo (podemos, en este caso, prescindir de los `\smallskip`). Como se quiere centrar *texto* en vez de símbolos matemáticos, hay que colocar el contenido dentro de un `\vbox` y desplegar éste. Consideramos, por ejemplo:

```
$$
\vbox{
\+ 1\quad &1\cr
\+ 1 & 2\quad &1 \cr
\+ 1 & 3 & 3\quad &1 \cr
\+ 1 & 4 & 6 & 4\quad &1\cr
}$$
```

y su resultado:

$$\vbox{\+ 1\quad &1\cr \+ 1 & 2\quad &1 \cr \+ 1 & 3 & 3\quad &1 \cr \+ 1 & 4 & 6 & 4\quad &1\cr }$$

Figura 8.2. Tabulación en un `\vbox` *centrado*

La regla exacta para colocar tabuladores es la siguiente: si se teclea una `&` a la derecha de todas las posiciones de tabulación existentes, se establece una nueva posición de tabulación tal que la columna recién completada tenga su anchura natural. Los `\quad` en el último ejemplo establecen sucesivas columnas de la anchura de un dígito más 1 em, dando el efecto visual de columnas con separación uniforme de 1 em.

8.2.2. Uso de renglones modelos.

Los tabuladores establecidos por la primera instancia de `\+...\cr` son apropiados solamente si el documento contiene un solo bloque de tabulación, o varios bloques de estructura idéntica. Para *redefinir* las posiciones de los tabuladores, TEX proporciona una segunda manera de colocarlos, con el comando `\settabs`.

Se teclea `\settabs\+`⟨*renglón modelo*⟩`\cr` para establecer las posiciones de los tabuladores como antes; pero el "renglón modelo" *no se imprime*. Esto es muy útil si no se sabe de antemano las anchuras apropiadas de las columnas: se determina la entrada más ancha en cada columna y con ellas se construya el renglón modelo. Luego las columnas que siguen serán alineadas apropiadamente. Ejemplo:

```
\settabs\+\indent&21--06\quad&Productos de papel e impresos\qquad&
    17.325,00&\quad&\cr
\+& 21--06& Productos de papel e impresos&\hfill\co 800,00&\cr
\+& 21--09& Materiales de oficina&\hfill 1.108,00&& (gastado)\cr
\+& 22--10& Adquisici\'on de libros& 17.325,00&& (en tr\'amite)\cr
```

que produce:

21–06	Productos de papel e impresos	¢800,00	
21–09	Materiales de oficina	1.108,00	(gastado)
22–10	Adquisición de libros	17.325,00	(en trámite)

Figura 8.3. Tabulación con un renglón modelo

donde la primera columna es una sangría (por `\indent`). La segunda columna del renglón modelo termina con `\quad` y la tercera con `\qquad`; así se garantiza la deseada separación mínima entre columnas. En la cuarta columna, se ajustan las cifras a la derecha usando el resorte `\hfill`—se supone que `\co` es un macro que produce el signo ¢ de colones. La quinta columna es un `\quad` de espacio, simplemente. La línea modelo termina con `&\cr`, pues no hace falta establecer la anchura de la sexta columna.

8.2.3. Columnas de igual anchura.

La tercera manera de colocar tabuladores es dividir la página en varias columna de igual anchura. Si se teclea `\settabs` n `\columns`, se producen renglones partidos en n columnas iguales. (Aquí no se requiere una línea modelo). Ejemplo:

```
\settabs 7 \columns
\+\it Presente& soy& eres& es& somos& sois& son \cr
\+\it Imperfecto& era& eras& era& \'eramos& erais& eran \cr
\+\it Pret\'erito& fui& fuiste& fue& fuimos& fuisteis& fueron \cr
\+\it Futuro& ser\'e& ser\'as& ser\'a& seremos& ser\'eis& ser\'an \cr
```

Este código produce:

Presente	soy	eres	es	somos	sois	son
Imperfecto	era	eras	era	éramos	erais	eran
Pretérito	fui	fuiste	fue	fuimos	fuisteis	fueron
Futuro	seré	serás	será	seremos	seréis	serán

Figura 8.4. Tabulación con columnas de igual anchura

8.2.4. Anulación de tabuladores.

Se puede quitar algunas tabuladores sin necesidad de redefinir todos, con `\cleartabs`. Este comando borra todos los tabuladores cuando ocurre fuera de un ambiente `\+...\cr`. Si se teclea `\cleartabs` dentro de una línea que empieza con `\+`, se borran solamente los tabuladores a la derecha de la columna que incluye `\cleartabs`. Esto es útil para alineamientos con tabulación variable, como son algunas programas de computación:

```
$$
\vbox{\cleartabs
\+\bf if $n < r$ &\bf then $n := n+1$\cr
\+ &\bf else &{\bf begin} {\it print\_totals\/}; $n := 0$;\cr
\+ && {\bf end};\cr
\+\bf while $p > 0$ do\cr
\+\quad\cleartabs &\bf begin & $q := {\it link}(p)$;\cr
\+ && {\it free\_node}$(p)$;\cr
\+ && $p := q$;\cr
\+ & {\bf end};\cr
}$$
```

cuyo resultado es:

```
if n < r then n := n + 1
         else begin print_totals; n := 0;
              end;
while p > 0 do
  begin q := link(p);
        free_node(p);
        p := q;
  end;
```

Figura 8.5. Tabuladores redefinidos con `\cleartabs`

Aquí, el primer `\cleartabs` elimina las posiciones previas de tabuladores, si existen; es una buena idea empezar una tabulación centrada con '`\vbox{\cleartabs...`' para eliminar las posiciones anteriores. Las primeras dos líneas tabuladas establecen nuevas posiciones para dos tabuladores. La cuarta línea usa solamente la primera columna; por ser más ancha que la columna establecida, proyecta a la derecha (sin generar mensajes de error). La quinta línea, que contiene `\cleartabs &`, borra los tabuladores y vuelve a establecer posiciones nuevas, de modo que la primera columna es un simple `\quad`. (Este ejemplo de fragmento de programa fue tomado del *TEXbook*, p. 234.)

8.3 Alineamiento con plantillas

Una alternativa más poderosa y flexible para alinear texto es el uso de *plantillas*. Se considera el alineamiento como una sucesión de columnas, donde las columnas individuales tienen características propias: algunas son centradas, otras justificadas a la izquierda o derecha; algunas usan letra negrilla, otras contienen fórmulas matemáticas solamente, etcétera. En vez de un "renglón modelo", este alineamiento debe tener un "preámbulo" en donde se establecen los formatos de cada columna y las separaciones entre ellas. Esta alternativa se obtiene con `\halign`, cuya sintaxis es:

`\halign{`⟨*preámbulo*⟩`\cr` ⟨*fila*$_1$⟩`\cr...` ⟨*fila*$_n$⟩`\cr}`

y las ⟨*fila*⟩s se imprimen según el formato establecido en el ⟨*preámbulo*⟩ que precede el primer `\cr`. Las filas en un `\halign` no usan la `\+`, que es más bien un recurso para activar tabuladores; pero sus columnas son demarcadas por signos `&` y cada fila debe terminar con `\cr`. El preámbulo se divide en columnas por el carácter `&` también; el contenido que cada columna se representa por la diesis `#` (sola, sin dígito), y además se colocan goma, resortes, cambios de estilo y otras cosas a su alrededor para establecer la "plantilla" de esa columna.

Por ejemplo, reconsideremos el segundo alineamiento de § 8.2:

```
$$\vbox{
\def\co{\rlap/c}
\halign{#\quad\hfil&#\quad\hfil&\hfil#\quad&#\hfil\cr
21--06& Productos de papel, cart\'on e impresos& \co 800,00\cr
21--09& Utiles y materiales de oficina& 1.108,00& (gastado)\cr
22--10& Adquisici\'on de libros& 17.325,00& (en tr\'amite)\cr
}}$$
```

usando `$$\vbox{...}$$` para centrar la tabla, y `\halign{...}` para controlar el formato. En el preámbulo, las primeras dos columnas, con plantilla `#\quad\hfil`, están justificadas a la izquierda, con un 'quad' al final para separarlas de lo que sigue; la tercera columna, con plantilla `\hfil#\quad`, está justificada *a la derecha*, con un 'quad' final; y la última, con plantilla `#\hfil`, está justificada a la izquierda. (El macro `\co` se define "localmente", dentro del `\vbox`: su definición desaparece cuando el `\vbox` termina).

Hay que incluir los `\quad` en la plantilla, por la forma en que obra el comando `\halign{...}`. Primero TEX lee todas las filas y determina cuál es la fila más ancha en cada columna; luego establece la anchura de cada columna con base en ese cálculo interno. (Es decir, lo que debe hacerse manualmente cuando se usa `\settabs\+`⟨*renglón modelo*⟩`\cr`, se hace en forma *automática* con el comando `\halign`.)

8.3.1. Plantillas con caracteres.

Cada columna del preámbulo establece una plantilla para la columna correspondiente de texto, y allí se pueden colocar otros caracteres también. Por ejemplo, en una tabla con números decimales, se puede usar

```
\halign{\indent#\hfil\quad&\hfil#&.#\hfil\quad&
        \hfil#&.#\hfil\cr ...}
```

para levantar la tabla siguiente*:

Producto Interno Bruto	230.2	100.0
a) Tradicionales	195.5	84.9
b) Materias Primas	14.6	6.3
c) Metal Mecánico	9.9	4.3
d) Residual	10.2	4.5

Figura 8.6. Alineamiento automático con `\halign`

* Ch. Zelaya *et. al.*, "Costa Rica Contemporánea", Editorial Costa Rica, San José, 1979.

Lo importante aquí es alinear los puntos decimales, y esto se hace insertándolos en las plantillas de las columnas en el preámbulo. En vez de tres columnas, hay cinco: la tercera y la quinta empiezan con '`.`' y no hay espacio para separarlas de las columnas anteriores. Se tecleó el alineamiento así:

```
\halign{\indent#\hfil\quad&\hfil#&.#\hfil\quad&\hfil#&.#\hfil\cr
\sl Producto Interno Bruto& 230&2 & 100&0\cr
a) Tradicionales& 195&5 & 84&9\cr
b) Materias Primas& 14&6 & 6&3\cr
c) Metal Mec\'anico& 9&9 & 4&3\cr
d) Residual& 10&2 & 4&5\cr}
```

y TEX se encargó de insertar los puntos decimales verticalmente alineados.

8.3.2. Entradas excepcionales.

Si una plantilla de columna es inadecuada para una o dos filas de la tabla, se puede empezar la columna de texto con `\omit`: esto *omite la plantilla* del preámbulo en la fila de marras y usa la plantilla "default" `#` en su lugar. Por ejemplo, para obtener 195,5 en vez de 195.5 arriba, se debe teclear:

```
a) Tradicionales& 195&\omit ,5\hfil & 84&9\cr
```

sin cambiar el tecleo de las otras filas. El punto desaparece al omitirlo de la plantilla; también hay que teclear `\hfil` para justificar la columna, porque ese resorte también fue omitido. Solamente la columna con `\omit` está afectada: en la quinta columna `9` produce .9, pues se usa la plantilla original '`.#\hfil`'.

8.4 Tablas regladas

8.4.1. Reglas horizontales en tablas.

Una regla horizontal entre dos filas de una tabla es una interrupción de la tabla; para obtenerla, se teclea

```
\noalign{\hrule}
```

entre las dos filas, i.e., después del `\cr` que termina la primera fila. Por ejemplo:

```
\halign{#\quad &#\hfil\quad &#\hfil\cr
a& b& c\cr
\noalign{\hrule}
u& v& w\cr
x& y& z\cr
```

a b c
u v w
x y z

Fíjese que la regla queda pegada a los renglones anterior y posterior: la interlineación usual en una pila de cajas no opera con reglas como `\hrule`. Hay dos salidas a este problema.

La primera solución usa el hecho de que *el argumento* de `\halign`, en cuanto contribución a una lista vertical de filas, obra *en el modo vertical*: `\noalign{\hrule}` es correcto, pero no se acepta `\noalign{\vrule}`. Los posibles argumentos de `\noalign` son reglas horizontales, cajas (horizontales o verticales), goma o resortes verticales, o penalties.

Entonces se puede colocar

```
\noalign{\smallskip\hrule\smallskip}
```

a b c
u v w
x y z

entre dos filas de un `\halign`.

La segunda solución es más usada. Plain TEX define el macro `\strut` ("puntal") como una caja con altura de 8.5 pt (algo más alto que una letra mayúscula), una hondura de 3.5 pt (más bajo que

caracteres como p, q, y) y una anchura de 0 pt: luego ¡es invisible! Colocamos un `\strut` en cada fila al insertar uno en el preámbulo:

```
\halign{\strut #\quad& #\quad& #\hfil\cr
```

y ahora la separación de las filas es más uniforme.

⟦ Se dispone también de un `\mathstrut`, que es una caja invisible de la altura y hondura de una paréntesis (a veces conviene incluirlo debajo del techo de un `\sqrt`). Un `\strut` es aproximadamente un punto más alto y un punto más hondo que un `\mathstrut`. ⟧

Para obtener una regla horizontal inicial o final, hay que encerrar el `\halign` en un `\vbox` y colocar un `\hrule` antes o después:

```
\vbox{\hrule\halign{...}\hrule}
```

Esto funciona porque un `\hrule` tiene la extensión horizontal de su caja vertical ambiente, que en este caso es la anchura total del `\halign{...}`.

8.4.2. Reglas verticales en tablas.

Para colocar reglas verticales entre dos columnas de una tabla, debemos considerar cada regla vertical como *una nueva columna* cuyo contenido es solamente esa regla. Por ejemplo:

```
\halign{\strut#\quad&\vrule#&\quad#&\quad#\cr
a&& b& c\cr
u&& v& w\cr
x&& y& z\cr}
```

a	b	c
u	v	w
x	y	z

Ahora tenemos *cuatro columnas*. Cada '`&&`' significa que la segunda columna no tiene texto de sustitución; luego su plantilla `\vrule#` es reemplazado por `\vrule` simplemente. (Cada plantilla del preámbulo debe tener un `#`; el código '`&\vrule&`' en el preámbulo generaría mensajes de error.) Esta regla tiene la altura del `\hbox` ambiente, el cual es la fila del alineamiento, cuya extensión vertical es determinado por `\strut`; luego `\vrule` da un puntal *visible*.

Obsérvese los pequeños quiebres entre los puntales sucesivos en este ejemplo: estos se deben a la goma `\lineskip` colocada automáticamente entre filas que están muy cercanas. Como los `\strut` separan sus contenidos, el `\lineskip` es en esta instancia un estorbo que debe suprimirse. Esto se logra con

```
\vbox{\offinterlineskip \halign{...}}
```

pues `\offinterlineskip` suprime el `\lineskip` dentro del `\vbox` ambiente. Así obtenemos

```
\vbox{\offinterlineskip
\halign{\strut#\quad&\vrule#&\quad#\quad&
        \vrule#&\quad#\cr
        a&& b&& c\cr
        u&& v&& w\cr
        x&& y&& z\cr}}
```

a	b	c
u	v	w
x	y	z

donde las reglas verticales se juntan para formar trazos verticales sin interrupción.

8.5 Diversos recursos para hacer tablas

8.5.1. Omisión de plantillas.

Si el contenido de alguna entrada de una tabla comienza con `\omit`, *se omite la plantilla* para esa entrada y se usa la plantilla simple `#` en su lugar. Esto permite sustituir materia con otro formato en esa entrada. Por ejemplo, si la primera fila de la tabla anterior se cambia en

```
a&&\omit \hfil bbb\hfil&& c\cr
```

se obtiene

a	bbb	c
u	v	w
x	y	z

Obsérvese los `\hfil`, dos resortes cuya función es centrar 'bbb', ya que la plantilla `\quad#\quad` fue cambiado a `#` por el `\omit`: hay que centrar esta entrada en forma explícita.

8.5.2. Fusión de columnas.

Es común que las tablas contengan filas excepcionales cuyas columnas desbordan los límites de las columnas de otras filas. Por ejemplo, un encabezado puede cruzar dos o tres columnas. Para fusionar n columnas de una sola fila, se teclea `\multispan` n al inicio de la primera y se omite los separadores `&`. Ejemplo: el código

```
\def\cc#1{\hfil\quad #1\quad\hfil}
\halign{\strut\hfil#\quad&\cc{#}&\cc{#}&\cc{#}&\cc{#}\cr
\#&\multispan3\hfil\sl Identificadores\hfil&\sl Nota\cr
1.& AA& BB& CC& 82\cr
2.& DD& EE& FF& 95\cr
3.& KK& LL& MM& 76\cr}
```

produce:

#	*Identificadores*			*Nota*
1.	AA	BB	CC	82
2.	DD	EE	FF	95
3.	KK	LL	MM	76

Figura 8.7. Alineamiento con algunas columnas fusionadas

Aquí se funden las columnas 2, 3 y 4 con `\multispan3`. En la primera fila se requieren dos `&` para separar las columnas 1, 2–3–4, 5; en las demás filas hay cuatro `&`. Un `\multispan` n, al igual que `\omit`, suprime el formato automático de las columnas fundidas; de ahí la necesidad de centrar el resultado con los resortes `\hfil`.

8.5.3. Plantillas repetidas.

El último ejemplo tuvo un preámbulo largo y repetitivo, aun con la definición del macro `\cc` para reducir el tecleo. Se obtiene el mismo resultado con:

```
\halign{\strut\hfil#\quad&&\hfil\quad#\quad\hfil\cr
```

Explicación: si un preámbulo es de la forma

⟨*parte*$_1$⟩ `&&` ⟨*parte*$_2$⟩ `\cr`

se repite la ⟨*parte*$_2$⟩ cuantas veces sean necesarias para llenar el número deseado de columnas. Es decir, el preámbulo es equivalente a:

⟨*parte*$_1$⟩ `&` ⟨*parte*$_2$⟩ `&` ⟨*parte*$_2$⟩ `& ... \cr`.

Si un preámbulo *comienza* con `&`, se repite todo el preámbulo periódicamente:

`&`⟨*bloque*⟩`\cr` ⟼ ⟨*bloque*⟩`&`⟨*bloque*⟩`&`⟨*bloque*⟩`& ...\cr`

8.5.4. Una tabla reglada completa.

Combinemos los diversos ingredientes en una tabla completa. La siguiente tabla:

Función	*Derivada*	*Primitiva*
x^n	nx^{n-1}	$\dfrac{x^{n+1}}{n+1}$
$\sin x$	$\cos x$	$-\cos x$
$\tan x$	$\sec^2 x$	$\ln\lvert\sec x\rvert$
$\dfrac{1}{1+x^2}$	$\dfrac{-2x}{(1+x^2)^2}$	$\arctan x$
$x^\alpha L_n^\alpha(x)$	$(\alpha+n)x^{\alpha-1}L_n^{\alpha-1}(x)$	$\dfrac{1}{\alpha+n+1}x^{\alpha+1}L_n^{\alpha+1}(x)$

Figura 8.8. Fórmulas en una tabla reglada

se obtiene con el código:

```
$$\vbox{\offinterlineskip
\def\a{\alpha}\def\cc#1{\hfil\quad#1\quad\hfil}
\hrule
\halign{&\vrule#&\strut\cc{$\displaystyle#$}\cr
height2pt&\omit&&\omit&&\omit&\cr
&\omit\cc{\sl Funci\'on}&&\omit\cc{\sl Derivada}&&
 \omit\cc{\sl Primitiva}&\cr
height2pt&\omit&&\omit&&\omit&\cr
\noalign{\hrule}
height2pt&\omit&&\omit&&\omit&\cr
& x^n    && nx^{n-1} && {x^{n+1} \over n+1} &\cr
& \sin x && \cos x   && -\cos x &\cr
& \tan x && \sec^2x  && \ln|\sec x| &\cr
& {1 \over 1+x^2} && {-2x \over (1+x^2)^2} &&\arctan x &\cr
& x^\a L_n^\a(x) && (\a+n) x^{\a-1} L_n^{\a-1}(x) &&
  {1 \over \a+n+1} x^{\a+1} L_n^{\a+1}(x) &\cr
height2pt&\omit&&\omit&&\omit&\cr
}\hrule}$$
```

Los `\omit\cc{...}` en la fila del encabezado sirven para suprimir el modo matemático de las columnas pares. Las filas terminan con `&\cr` y no sólo con `\cr`: esto genera una séptima columna, el trazo vertical del borde derecho de la tabla. Las líneas `height2pt&\omit&&\omit&&\omit&\cr` generan una separación textual de 2 pt para destacar el encabezado. En estas líneas los `\omit` suprimen los puntales `\strut`; la altura de la fila es 2 pt, ya que en la primera columna la plantilla `\vrule#` es reemplazado por `\vrule height 2pt`; en las demás columnas impares, el contenido vacío expande `\vrule#` en `\vrule`, lo cual genera una regla de la altura de la fila, que ya está establecida en 2 pt. Luego, los trazos verticales entre las fórmulas no se interrumpen.

8.6 Resortes en las tablas

Las entradas en una tabla se pueden justificar con resortes `\hfil` en el preámbulo o bien en las entradas individuales. Si una columna tiene `\hfil...#...\hfil` en el preámbulo, hay que usar el resorte más potente `\hfill` en vez de `\hfil` para justificar una sola entrada a la derecha.

Se obtiene una tabla de una anchura predeterminada, como 10 cm, con `\halign to 10cm{...}`. Los resortes en las columnas se estirarán para justificar columnas individuales; pero, si las anchuras de éstas suman menos de 10 cm, los resortes *dentro* de cada columna no podrán estirarse para cubrir la anchura faltante. Por eso, TEX coloca una goma especial llamado `\tabskip` *entre las columnas* de una tabla, y también antes de la primera columna y después de la última. Su valor usual es de 0 pt, pero `\tabskip =` ⟨*nuevo valor*⟩ lo cambiará a gusto. Ejemplo:

```
\vbox{\tabskip=20pt
\halign{#&#&#&#\cr
a & b & c & d\cr
xx & yyy & zzzz & w\cr}}
```

a	b	c	d
xx	yyy	zzzz	w

TEX decide el monto de goma que coloca entre las columnas al notar el valor de `\tabskip` (i) cuando lee la llave `{` que comienza el preámbulo; (ii) cuando lee cada `&` del preámbulo que separa dos columnas; (iii) cuando lee el `\cr` que termina el preámbulo. Por ende, el código:

```
\tabskip = 1em
\halign{#\hfil\tabskip=3em&#&\tabskip=4em\bf#&#\tabskip=0em\cr
```

establece una tabla con goma de 1 em antes de la primera columna, goma de 3 em entre las columnas 1 y 2 y también entre las columnas 2 y 3, goma de 4 em entre las columnas 3 y 4, y ninguna goma después de la última columna. Las reasignaciones de `\tabskip` *no forman parte de las plantillas* de las columnas, que son '`#\hfil`', '`#`', '`\bf#`', '`#`' respectivamente.

9 Rutinas de Salida

Cuando TEX decide que el texto acumulado es suficiente para formar una página, interrumpe la lectura de texto, confecciona la página en su forma final y la envía a disco. En la terminal (en el archivo 'TeX log') aparece un corchete `[` seguido por el número de página; cuando la página está completa, aparece otro corchete `]` y el programa vuelve a leer materia para la página siguiente.

La página así confeccionada es parte de la *Lista Vertical Principal*, que consta de los renglones de párrafos ya levantadas, las gotas de goma y los resortes que separan estos renglones, y algunos otras cosas como reglas o cajas colocadas explícitamente entre párrafos. Si el último párrafo leído de la página actual desborda esta página, TEX guarda una parte para la página siguiente, y luego decide cómo "secar" la goma vertical en la página actual; hecho ésto, envía esta página a disco y procede con la página que sigue.

Si una página consta de texto solamente, la única flexibilidad vertical está en la goma que se coloca entre los renglones y los párrafos. Pero si hay secciones divididos por `\smallskip` o `\medskip` o `\bigskip`, o si hay despliegues matemáticos, estos incorporan goma vertical contractible y estirable: en este caso, los cortes de página son usualmente satisfactorias.

Si una página se cortara en un lugar inconveniente en un primer intento, hay dos opciones para volver a cortar la página en un lugar predeterminado. El comando `\eject` ("expulse") significa "corte la página aquí y envíela a disco". Al cortar de esta forma, la goma vertical en la página se estira de tal manera que el último renglón quede al fondo de la página; y si se estira mucho, se genera una protesta sobre un '`Underfull \vbox while output is active`'. La segunda opción, más común, es la de rellenar el remanente de la página con espacio en blanco: ésto se hace con `\vfill\eject`.

A la hora de enviar la página a disco, se colocan los números de página, los encabezados, las notas al pie y las inserciones. Este proceso de agregar las añadiduras de oficio se llama la *rutina de salida*.

9.1 La rutina de salida de Plain TEX

La rutina usual de Plain TEX supone que se usan hojas de papel de 8.5 in × 11 in (tamaño "Carta US"). Hay un margen por defecto de 1 in a los cuatro lados. (Luego el área para impresión

es de 6.5 in × 9 in). Esta área se modifica al cambiar los dos parámetros `\hsize` y `\vsize`. En Plain TEX, se usan `\hsize=6.5in` y `\vsize=8.9in`. Si se cambian estos parámetros (por ejemplo, podríamos pedir `\hsize=34pc` y `\vsize=48pc`), el margen superior y el margen izquierdo seguirán siendo de 1 in, mientras los márgenes derecho e inferior cambiarán.

Para cambiar los márgenes superior e izquierdo, se puede mover todo el texto de la página hacia la derecha con `\hoffset` y hacia abajo con `\voffset`. Por ejemplo, `\hoffset=1pc \voffset=-3pt` mueve el texto 1 pica hacia la derecha y 3 puntos hacia arriba (−3 pt hacia abajo es +3 pt en la dirección contraria). Estos cambios a `\hsize`, `\vsize`, `\hoffset`, `\voffset`, deben hacerse *al inicio* de un documento.

Para quitar los números de página que automáticamente aparecen centrados al pie, se puede teclear `\nopagenumbers` al inicio del documento.

La rutina de salida de Plain TEX funciona del siguiente modo. Cuando TEX decide cortar una página, envía el contenido de la página, en una caja, a una salita de espera especial. Luego busca una lista de instrucciones de salida llamado `\output`, el cual es un parámetro especial de TEX cuyo valor es una *lista de fichas*. En Plain TEX, se pone `\output = {\plainoutput}`, donde `\plainoutput` es un macro que se expande en:

```
\shipout\vbox{\makeheadline\pagebody\makefootline}
\advancepageno
\ifnum\outputpenalty>-20000 \else\dosupereject\fi
```

Aquí, `\shipout` es el comando que envía una caja (en este caso, la página completa) a disco, y la borra de la memoria activa para que TEX pueda continuar con la página que sigue. (Posteriormente, el programa que confecciona la imagen en pantalla —esta función viene integrada en *Textures*, pero en otras implementaciones se trata de un programa externo— la buscará en el disco). Luego, se incrementa el número de página con `\advancepageno`; y la instrucción condicional se invoca si esta página es la última y si quedan inserciones que todavía no hayan salido.

Una página consta entonces de cabeza, cuerpo y pie, los cuales se ordenan con la separación correcta usando `\makeheadline`, que coloca una caja con la `\headline` al inicio; `\pagebody`, que confecciona la página del material en la salita de espera con inserciones y notas al pie incluidas; y `\makefootline`, que coloca una caja con la `\footline` al final.

9.2 Encabezados y pies

9.2.1. Colocación de encabezados y pies.

Examinemos la definición de `\makeheadline` en Plain TEX:

```
\def\makeheadline{\vbox to 0pt{\vskip-22.5pt
        \line{\vbox to 8.5pt{}\the\headline}\vss}\nointerlineskip}
```

y la de `\makefootline`:

```
\def\makefootline{\baselineskip=24pt \line{\the\footline}}
```

Los parámetros `\headline` y `\footline` son también listas de fichas. El comando `\the` es un comando primitivo de TEX que convierte la forma interna de una lista de fichas en una cadena de caracteres (es decir, "desempaca" la representación interna del parámetro que sigue). La caja vacía de altura 8.5 pt es un puntal para uniformar las alturas de los diversos encabezados posibles. El `\baselineskip=24pt` es un artificio para separar el pie del cuerpo de la página, creando el equivalente de una línea en blanco (pues el valor usual de `\baselineskip` es 12 pt); las instrucciones de la rutina de salida se efectúan localmente y la interlineación de la página siguiente no sufre modificación.

¿Por qué `\vskip-22.5pt`? Resulta que TEX coloca una interlineación especial `\topskip` al inicio de cada página, para establecer una distancia uniforme entre el techo de la página y la base

del primer renglón. En Plain TEX, se tiene `\topskip=10pt`. Para que el encabezado esté separado por una línea en blanco del primer renglón de la página, hay que retroceder (con un `\vskip` de una longitud negativa) por un monto de

$$2 \times 24\,\mathrm{pt} + 8.5\,\mathrm{pt} - 10\,\mathrm{pt} = 22.5\,\mathrm{pt}.$$

El `\nointerlineskip` después de la caja del encabezado suprime la interlineación usual, para no destruir el resultado de este cálculo.

Finalmente, se esconde la altura del encabezado al colocarlo dentro de un `\vbox to 0pt`, seguido por el resorte contractible `\vss` para que no haya protesta sobre un `Overfull \vbox`. En general, la construcción `\vbox to 0pt{`⟨*contenido*⟩`\vss}` crea una caja vertical sin mover el punto de referencia: el contenido entonces puede traslapar el texto que sigue.

9.2.2. Modificación de los encabezados y pies usuales.

El valor normal de `\headline` es `{\hfil}`, un resorte invisible, que da una línea en blanco. El `\footline` es `{\hss\tenrm\folio\hss}` en Plain TEX: esto coloca el número de página (con `\folio`) al final, centrado en la fuente romana de 10 puntos. Si ahora se pretende cambiar el formato de página para modificar los encabezados y los pies, es suficiente cambiar `\headline` y `\footline`. Para poner un encabezado con los números de página arriba, con el pie en blanco, se recomienda colocar (al inicio del archivo) lo siguiente:

```
\nopagenumbers
\headline={\ifodd\pageno\dercabeza\else\izqcabeza\fi}
\def\dercabeza{\hfil\sevenrm\titulito\hfil\tenrm\folio}
\del\izqcabeza{\tenrm\folio\hfil\sevenrm\autorcito\hfil}
\voffset=2\baselineskip
```

El `\nopagenumbers` es un macro que significa `\footline={\hfil}`: luego se elimina el pie de página. Con el `\voffset`, se reacomoda la página hacia abajo para dejar campo para el encabezado.

Falta definir `\autorcito` y `\titulito`, que serán específicas a cada documento. Por ejemplo, `\def\autorcito{PEDRO PEREZ PEREIRA}` y `\def\titulito{POBRE PINTOR PORTUGUES}` serán aptas para la autobiografía de aquél peregrino que pasó por París pintando preciosos paisajes.

9.2.3. Números arábigos y romanos.

Las páginas iniciales de un libro a menudo se indican con números romanos: v, vi, vii, etcétera. Para esas páginas, `\pageno` debe ser *negativo*. (Para empezar un prólogo en la página v, se coloca `\pageno=-5` al inicio del archivo.) El macro `\folio` se define así:

```
\def\folio{\ifnum\pageno<0 \romannumeral-\pageno
                \else \number\pageno \fi}
```

(Aquí, `\romannumeral` imprime un número† con letras romanas; mientras `\number` lo imprime con dígitos arábigos. Este año, por ejemplo, `\number\year` imprime '2008', pero `\romannumeral\year` produce 'mmviii'; `\year` es la representación del año que mantiene el sistema operativo.) Al terminar la página, el comando `\advancepageno` suma 1 a `\pageno` si éste es positivo, pero resta 1 si `\pageno` es negativo: lo que sigue a la página viii es la página ix, no la página vii. Al llegar a la parte principal del libro, se dice `\pageno = 1` para empezar el conteo con números arábigos.

† Mejor dicho, la representación interna de un número; aquí conviene distinguir entre un número representado por una configuración de un registro de memoria de la computadora, y su presentación convencional mediante caracteres del teclado.

9.3 Formato de doble columna

Hay varias maneras de obtener un formato de dos columnas. La manera más simple es de considerar las dos columnas como páginas independientes, que no se imprimen por separado sino juntos. Para poder juntarlos, hay que guardar la columna izquierda en reserva hasta que la columna derecha esté lista. Se aprovecha la metodología de TEX para poder hacer esta manipulación. Lo que hay que recordar es que (i) cuando TEX tiene suficiente materia para una página, lo que hace es leer la lista de fichas `\output`; (ii) solamente el comando `\shipout` es capaz de enviar materia a disco. Cuando se desea formato de dos columnas, asignamos `\output = {\doblecolumna}` (¡no olvide las llaves!); podemos regresar al formato usual con `\output = {\plainoutput}` más tarde.

⟦Para definir `\doblecolumna`, hay que anticipar algunos convenios de asignación que discutiremos más adelante. Se puede reservar una de las cajas internas accesibles en TEX con el macro `\newbox`⟨*caja*⟩, y llenarla con el código

`\setbox`⟨*caja*⟩ `= \hbox{`⟨*contenido*⟩`}`, o bien
`\setbox`⟨*caja*⟩ `= \vbox{`⟨*contenido*⟩`}`.

Aquí ⟨*caja*⟩ es una palabra de control que identifica la caja reservada. También se puede asignar una nueva condicional con `\newif\ifalgo` donde `\ifalgo` es en general una palabra de control que comienza con '`\if`'; sus valores de verdadero y falso se denotan `\algotrue` y `\algofalse`. Finalmente, una dimensión nueva recibe un nombre reservado con `\newdimen`.⟧

Preparamos el formato de dos columnas con

```
\newbox\colizqda
\newdimen\anchurareal  \anchurareal=\hsize
\hsize=0.47\hsize
\newif\ifderecha   \derechafalse
```

La caja `\colizqda` recibirá el contenido de la columna izquierda. La longitud `\anchurareal` conserva el valor del `\hsize` original, y la asignación `\hsize=0.47\hsize` *modifica* la anchura a una medida apropiada para una sola columna. (Sobra una longitud de $6\% = 100\% - 2 \times 47\%$ de la anchura original de la página, que se usará para separar las dos columnas. Es necesario modificar el `\hsize` porque TEX *confecciona sus renglones de párrafos en cajas de anchura* `\hsize`. La condicional `\ifderecha` señala la columna actual (derecha o izquierda); inicialmente la condición es *falsa*, pues se empieza con la columna izquierda.

Después de estos preparativos, establecemos la rutina de salida:

```
\def\doblecolumna{\ifderecha
      \shipout\vbox{\makelongheadline
                    \hbox to \anchurareal{%
                      \box\colizqda \hfil
                      \leftline{\pagebody}}
                    \makelongfootline}
      \global\derechafalse
      \advancepageno
    \else
      \global\setbox\colizqda=\leftline{\pagebody}
      \global\derechatrue
    \fi
    \ifnum\outputpenalty>-20000 \else\dosupereject\fi}
```

La última línea es idéntica a la de `\plainoutput`. Consideramos lo que ocurre cuando hay suficiente materia para la primera columna izquierda, es decir, cuando la longitud de la lista vertical principal excede `\vsize`. Se llama `\output`, que causa la lectura de `\doblecolumna`. Como inicialmente vale `\derechafalse`, se ignora todo antes del `\else` y se asigna la materio de la primera

columna, ya preparada en `\pagebody` como página delgada, a la caja `\colizqda`. La función de `\leftline#1` (cuya definición es `\hbox to \hsize{#1\hss}`) es de empacar la "caja vertical" `\pagebody` en una "caja horizontal". Luego se cambia la señal de columna al decir `\derechatrue`, se ejecuta la última línea y se regresa a leer más texto. La columna izquierda ha sido confeccionada y guardada, pero *no ha sido enviada a disco.*

Después de un rato, habrá suficiente materia nueva como para exceder `\vsize` otra vez. De nuevo se lee `\doblecolumna`, pero ahora vale `\derechatrue` y se usa la primera parte del macro condicional `\ifderecha...\else...\fi`. Ahora se ejecuta un `\shipout`, enviando a disco una caja vertical que contiene tres cajas empiladas. El `\makelongheadline` y `\makelongfootline` son análogas a `\makeheadline` y `\makefootline`, pero deben crear cajas de anchura `\anchurareal`. En el medio hay una caja horizontal con dos cajas adentro: la `\colizqda`, que contiene materia para la columna izquierda, y el nuevo `\pagebody`, que será la columna derecha. Entre ellos se coloca un resorte `\hfil`, para llenar el 6% residual de `\anchurareal` y así separar las columnas. Enseguida se cambia la señal a `\derechafalse` para empezar de nuevo en la columna izquierda y se incrementa el número de la página. Ahora se puede repetir el ciclo y confeccionar las dos columnas de la página siguiente.

10 Registros

10.1 Los registros de TEX

Hasta ahora, la discusión del funcionamiento de TEX ha asumido que los comandos de TEX siempre producen los efectos deseados, sin dar importancia a la forma en que logran producir dichos efectos. Echaremos ahora un vistazo al manejo interno de TEX, y a los lugares donde se guarda información acerca de sus parámetros y cajas, que permite que trabaja de modo eficiente.* Antes de empezar a levantar texto, TEX toma un buen trozo de memoria RAM y lo divide en "registros" de 4 bytes c/u; hay 256 registros cada uno para: a) números enteros, b) longitudes, c) gotas de goma, d) gotas de una goma especial para fórmulas, e) listas de fichas y f) cajas.

Nombre del registro	*Tipo de registro*	*Asignación del registro*
`\count`	número entero	`\count` n = ⟨*entero*⟩
`\dimen`	longitud	`\dimen` n = ⟨*dimensión*⟩
`\skip`	goma	`\skip` n = ⟨*goma*⟩
`\muskip`	"mugoma"	`\muskip` n = ⟨*mugoma*⟩
`\toks`	lista de fichas	`\toks` n = {⟨*fichas*⟩}
`\box`	caja	`\setbox` n = ⟨*caja*⟩

Figura 10.1. Clasificación de los registros de TEX

Aquí n toma los valores $0, 1, 2, \ldots, 255$. Es decir, los registros numéricos están enumerados de `\count0` a `\count255`; los registros de dimensión van de `\dimen0` a `\dimen255`; los registros de goma ordinaria y matemática de `\skip0` a `\skip255` y de `\muskip0` a `\muskip255`; los registros para fichas de `\toks0` a `\toks255`; y los registros de cajas de `\box0` a `\box255`.

La "mugoma" es una especie de goma especial que se usa en el modo matemático solamente, y se mide en `mu` ("math unit") en vez de `pt`: el tamaño del `mu` depende del tamaño de letra ambiente; de hecho, 18 mu = 1 em. El "medio espacio" `\,` es en realidad `\mskip 3mu`. El comando `\mskip` coloca "mugoma" en el modo matemático, por analogía con `\hskip` y `\vskip` en los otros modos.

La sintaxis de las asignaciones se resume en la tabla que sigue; una barra vertical separa dos formas alternativas de sustituir el lado izquierdo, y las paréntesis rodean cosas opcionales.

* Los que quieren saber la historia completa deben consultar el libro *TEX: The Program*, que contiene la fuente Pascal del programa, completamente documentado.

⟨*entero*⟩	⟼	un número entero
⟨*dimensión*⟩	⟼	⟨*número*⟩ ⟨*unidad*⟩
⟨*goma*⟩	⟼	⟨*gota*⟩ (`plus` ⟨*gota*⟩) (`minus` ⟨*gota*⟩)
⟨*mugota*⟩	⟼	⟨*número*⟩ `mu`
⟨*fichas*⟩	⟼	una lista de caracteres y comandos
⟨*caja*⟩	⟼	⟨*pedido de caja*⟩`{`⟨*fichas*⟩`}`
⟨*número*⟩	⟼	un número entero o decimal
⟨*unidad*⟩	⟼	`pt` | `pc` | `in` | `bp` | `cm` | `mm` | `dd` | `cc` | `sp` | `em` | `ex`
⟨*gota*⟩	⟼	⟨*dimensión*⟩ | ⟨*número*⟩ `fil`
		| ⟨*número*⟩ `fill` | ⟨*número*⟩ `filll`
⟨*mugoma*⟩	⟼	⟨*mugota*⟩ (`plus` ⟨*mugota*⟩) (`minus` ⟨*mugota*⟩)
⟨*pedido de caja*⟩	⟼	⟨*nombre de caja*⟩ | ⟨*nombre de caja*⟩ `to` ⟨*dimensión*⟩
		| ⟨*nombre de caja*⟩ `spread` ⟨*dimensión*⟩
⟨*nombre de caja*⟩	⟼	`\hbox` | `\vbox` | `\vtop` | `\vcenter`

Figura 10.2. Convenios para los contenidos de los registros de TEX

Por ejemplo, las siguientes asignaciones son legales:

```
\count14 = -4096
\dimen100 = 13,5 pc
\skip7 = 10pt plus 1 fil minus 0.5fil
\muskip222 = 4 mu plus 2mu minus 4mu
\toks30 = {Una frase \hfil\ partida en 2}
\setbox0 = \hbox to 0pt{$\Rightarrow$\hss}
```

Algunos registros son usados por TEX para sus propias operaciones. Por ejemplo, el número de la página es el contenido del registro `\count0`. Plain TEX define el sinónimo `\pageno` para este registro. Así, si un trabajo debe comenzar con las páginas numeradas de 85 en adelante, se declara

`\pageno = 85` o equivalentemente `\count0 = 85`

al inicio del archivo fuente.

Los registros `\count1`, ..., `\count9` son registros reservados; en Plain TEX normalmente valen 0. El registro de cajas `\box255` es muy especial: este es el lugar (la "salita de espera") donde se guarda el contenido de la página actual: ¡no modifique esta caja sin estar absolutamente seguro de lo que usted hace! Si hay notas al pie, se guardan en `\box254`, y otras inserciones usan `\box253`, `\box252`, etc. Para manejo temporal de cajas, se recomienda usar `\box0`, `\box1`, ..., `\box9` solamente.

10.2 Designación de registros

Para evitar posibles conflictos entre macros que modifican un mismo registro (`\dimen40`, digamos) independientemente, Plain TEX provee un método de reservar ciertos registros para uso privado: es cuestión de designar el registro por un *nombre* (una palabra de control) y pedirle a TEX que reparte el primer registro del tipo deseado que aún no está en uso. La instrucción

```
\newcount\lacuentaporfavor
```

define el comando `\lacuentaporfavor` como sinónimo de `\count` m donde $(m-1)$ es el número del registro `\count` más recientemente designado. (Plain TEX inicialmente reserva `\count0`, ..., `\count25`; la primera instrucción `\newcount` reservará el registro `\count26`, y así sucesivamente.) Luego se puede asignar un valor al registro en la forma usual:

```
\lacuentaporfavor = 666
```

De forma análoga, hay instrucciones `\newdimen`, `\newskip`, `\newmuskip`, `\newtoks` y `\newbox` para reservar registros de los otros tipos. Ya hemos ejemplificado estas asignaciones en la § 9.3, en la discusión de formatos de doble columna.

⟦ Plain TEX también proporciona `\newif` para crear condicionales reservados: vea la declaración de `\ifderecha` en la § 9.3. Sin embargo, no hay registros para almacenar condicionales; el mecanismo de repartición de `\newif` es un tanto diferente de los otros: vea el *TEXbook*, pp. 347–348. ⟧

10.3 Operaciones aritméticas con registros

TEX tiene cierta capacidad de efectuar operaciones aritméticas con los registros numéricos. Puede sumar a un registro un valor del tipo apropiado, y puede multiplicar (o dividir) el valor de un registro por un número entero. Los comandos apropiados son:

`\advance\count` n `by` ⟨*entero*⟩
`\advance\dimen` n `by` ⟨*dimensión*⟩
`\advance\skip` n `by` ⟨*goma*⟩
`\advance\muskip` n `by` ⟨*mugoma*⟩
`\multiply`⟨*registro*⟩ `by` ⟨*entero*⟩
`\divide`⟨*registro*⟩ `by` ⟨*entero*⟩

donde ⟨*registro*⟩ es un registro de tipo `\count`, `\dimen`, `\skip` o `\muskip`.

Las siguientes operaciones son legales:

```
\advance\pageno by 1
\advance\hsize by 1pc
\advance\dimen40 by \count28pc
\smallskipamount = 3pt plus 1pt minus 1pt
\advance\parskip by \smallskipamount
```

El parámetro `\smallskipamount` ha sido creado por Plain TEX (usando `\newskip`) y la asignación aquí dada es lo que Plain TEX hace. (El comando `\smallskip` es un macro cuya expansión es `\vskip\smallskipamount`). La operación con `\parskip` tiene el efecto de colocar un `\smallskip` después de cada párrafo en forma automática.

La instrucción `\divide` hace división entera, sin guardar residuos. Así,

```
\count2 = 17  \divide\count2 by 3
```

es equivalente a `\count2 = 5`.

10.3.1. Aritmética con dimensiones de cajas.

Si una caja está guardada en el registro `\box` n, su altura, hondura y anchura se guardan como `\ht` n, `\dp` n y `\wd` n respectivamente. Para hacer una operación aritmética con estas dimensiones, hay que usar un registro `\dimen` disponible. Los registros `\dimen0`, ..., `\dimen9` se mantienen libres para este tipo de cálculos. Así, el código

```
\newbox\micaja  \newdimen\arriba
\setbox\micaja = \hbox{$\displaystyle\sum_0^\infty$}
\dimen0 = \ht\micaja
\advance\dimen0 by \dp\micaja
\divide\dimen0 by 2
\arriba = \dimen0
```

calcula la proyección de una fórmula por encima de su eje central. (TEX hace un cálculo análogo internamente cada vez que emplea `\left...\right`). Si h es la altura de la fórmula y d su hondura, es cuestión de calcular $(h + d)/2$, y archivar el resultado en el registro `\arriba`.

10.4 Impresión del contenido de un registro

Un registro `\toks` almacena una lista de caracteres y comandos; pero un registro `\count` guarda su contenido — un número entero — en un formato binario inaccesible. A veces se requiere "desempacar" un registro `\count`, transformando su contenido en el juego de dígitos arábigos que lo representa. Para este efecto, TEX tiene una operación `\the` que transforma el contenido de un registro en una lista de caracteres ASCII.

Consideremos algunos ejemplos:

`\pageno=40 \advance\pageno by 2`	`\the\pageno` $\longmapsto$ `42`
`\hsize = 34pc`	`\the\hsize` $\longmapsto$ `408.0pt`
`\skip3 = 1pc plus 3pt`	`\the\skip3` $\longmapsto$ `12.0pt plus 3.0pt`
`\everypar={\leftskip=\parindent}`	`\the\everypar` $\longmapsto$ `\leftskip=\parindent`

En un registro de tipo `\dimen`, `\the` produce el valor en puntos `pt`. (Luego $1\,\mathrm{pc} \equiv 12.0\,\mathrm{pt} \implies 34\,\mathrm{pc} \equiv 408.0\,\mathrm{pt}$). El comando `\everypar` es un registro interno de tipo `\toks`, que TEX lee al inicio de cada párrafo. Normalmente está vacío, pero siempre está disponible para formatear una serie de párrafos de manera uniforme. La expansión de `\the\everypar` es una copia de la lista de fichas que TEX procesa como si los hubiera recién leído en el archivo fuente.

El contenido de un registro se despliega en la terminal (en el archivo TeX log) si el código

`\showthe`⟨*registro*⟩

está en el archivo fuente. Por ejemplo,

`\showthe\overfullrule` produce `> 5.0 pt .` en el archivo TEX log.

El `\overfullrule` es la anchura de la caja negra que aparece a la derecha de una caja horizontal desbordada; en Plain TEX, su valor es de 5 pt (para que el autor vea su error y lo corrija!)†

10.4.1. Conversión de caja de letra.

Siguiendo una tradición establecida en el taller de Gutenberg en Mainz, desde 1450, las letras de una fuente son unas piezas metálicas guardadas en una caja abierta que tenía dos partes: en la "caja alta" se guardan las letras mayúsculas y en la "caja baja" las minúsculas. En el último medio siglo, con el auge de las tecnologías de fotocomposición y levantamiento digital de texto, las piezas metálicas han caído en desuso. TEX tiene dos comandos que cambian minúsculas a mayúsculas y viceversa, y *actúan sobre listas de fichas*. El comando

`\uppercase{`⟨*lista de fichas*⟩`}`

convierte cada carácter de letra en la lista en su equivalente mayúscula, sin cambiar otros caracteres ni macros. También,

`\lowercase{`⟨*lista de fichas*⟩`}`

convierte cada carácter de letra de la lista en su equivalente minúscula. Ejemplo:

`\uppercase{!‘Una \quad u\~na!}` $\longmapsto$ `!‘UNA \quad U\~NA!`

† En LATEX, `\overfullrule` vale 0 pt; esta mala práctica puede remediarse al usar la opción '`draft`' con el `\documentclass`.

11 Manejo de cajas

11.1 Cajas empiladas y enfiladas

11.1.1. Cajas verticales.

TEX tiene tres tipos de cajas verticales: `\vbox`, `\vtop` y `\vcenter`. El tercer tipo funciona *únicamente en el modo matemático*, donde es útil para centrar fórmulas (por ejemplo, `\eqalign` crea una caja `\vcenter` para guardar su argumento). Un `\vbox` o `\vtop` puede usarse en cualquier modo. El contenido de cualquiera de estos tres tipos de caja es una *lista vertical* de cajas, reglas, goma y resortes; entre dos cajas consecutivas de la lista se coloca la goma `\lineskip`, como si fueran renglones de un párrafo.

La diferencia entre las tres cajas verticales es la posición de la línea de base. Si formamos una pila de cajas en un `\vbox`, la hondura del `\vbox` es la hondura de la caja más baja de la pila. Si empilamos cajas en un `\vtop`, la altura del `\vtop` es la altura de la caja más alta de la pila. En un `\vcenter`, la altura y la hondura son iguales, no importa cuál sea su contenido. La altura más la hondura dan la extensión vertical total de la caja.

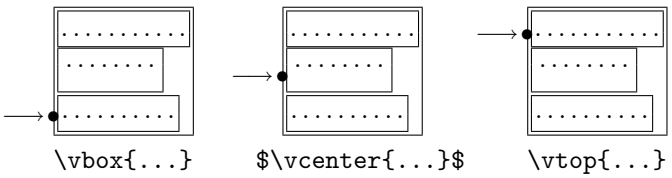

Figura 11.1. Puntos de referencia de las cajas verticales

11.1.2. Cajas horizontales.

Sólo hay un tipo de caja horizontal: `\hbox{...}`. Su contenido es una *lista horizontal* de cajas, reglas, goma y resortes, enfilados de izquierda a derecha. Podemos pedir la anchura explícitamente con `\hbox to` ⟨*dimensión*⟩`{...}`. Por ejemplo, `\line{...}` es simplemente `\hbox to \hsize{...}`. Cada renglón de un párrafo se guarda en un `\line` una vez que TEX haya leído el párrafo completo y lo haya dividido en renglones.

Se puede subir o bajar cualquier caja horizontal `\hbox` por una longitud determinada. Es así como se obtiene la letra E en TEX, deliberadamente más bajo que lo usual: se encierra la E en una caja y se desplaza esta caja hacia abajo. Por ejemplo, para obtener el logo TEX, se puede teclear:

```
T\kern-.1667em\lower.5ex\hbox{E}\kern.125em X
```

Ahora, .1667 $\doteq \frac{1}{6}$ y .125 $= \frac{1}{8}$; un `\kern-.1667em` es un movimiento de $\frac{1}{6}$ em hacia atrás: un pequeño retroceso. La letra E se envuelve en una caja, y esta caja se desplaza $\frac{1}{2}$ ex hacia abajo con `\lower.5ex\hbox{E}`. Luego la receta para el logo es: "una letra T, un retroceso de $\frac{1}{6}$ em, una letra E desplazada hacia abajo por $\frac{1}{2}$ ex, un retroceso de $\frac{1}{8}$ em y una letra X".

11.1.3. Listas verticales y horizontales.

Es hora de hacer un inventario completo de los ingredientes que pueden forman parte de una lista vertical. En una pila podemos colocar:

- ◇ una caja (vertical u horizontal);
- ◇ una regla (`\hrule`);
- ◇ una gota de goma o un resorte;
- ◇ un crenaje (`\kern`⟨*dimensión*⟩);
- ◇ una inserción (como `\footnote` o `\midinsert` o `\topinsert`);
- ◇ un penalty (`\penalty`⟨*entero*⟩);
- ◇ una marca (con `\mark`);
- ◇ un quéeseso ("whatsit": con `\write` o `\special`).

Los últimos dos ítemes son cosas especiales que veremos luego. Un "penalty" en la lista vertical principal baja (o, si es negativo, sube) la probabilidad de un corte de página; no tiene efecto notable dentro de una caja vertical. En una caja vertical, un crenaje ejecuta un movimiento hacia abajo del monto especificado (o hacia arriba, si este monto es una longitud negativa); en una caja horizontal, un crenaje produce un desplazamiento hacia la derecha o la izquierda.

Hay reglas precisas para los posibles ingredientes de una "lista horizontal", es decir, un párrafo o el interior de un `\hbox`, y también para una "lista matemática" formado por una fórmula en el modo matemático: vea el *TEXbook*, pp. 95 y 157. En general, las listas contienen una sucesión de cajas separadas a veces por goma, resortes y crenajes, con algunos adornos como reglas, penalties, marcas, etc., intercalados de vez en cuando.

11.2 Párrafos en cajas verticales

Si un `\vbox` o `\vtop` contiene caracteres, TEX confeccionará un párrafo una vez que haya leído el contenido (o antes, si encuentra una línea en blanco o un `\par` dentro de la caja). Es importante recordar que un párrafo empieza con un `\hbox` vacío de anchura `\parindent` (para la sangría) y que sus renglones se empacan en cajas horizontales de anchura `\hsize`. Luego, la anchura del `\vbox` será `\hsize` si su contenido incluye un carácter — aunque sea sólo uno. Si así resultara muy ancho, hay dos alternativas:

a) si se desea un solo renglón corto de anchura 10 pc (digamos), se puede empacar los caracteres dentro de otro `\hbox`, porque TEX no empieza un párrafo cuando encuentra una caja explícita:

```
\vbox{\hbox{Un ejemplo de una caja dentro de otra}}
```

b) si el contenido desbordara la anchura deseada, se podría cambiar diversos parámetros de página dentro del `\vbox` (los valores originales se restaurarán al salir de la caja):

```
\vbox{\hsize=10pc \parindent=0pt \pretolerance=300 \tolerance=600
      Un p\'arrafo largo ... con bastante texto.}
```

Ahora el texto será justificado en renglones de anchura 10 pc, sin sangría y con más "tolerancia" para renglones con espacios estirados entre palabras. (Plain TEX coloca `\pretolerance=100` y `\tolerance=200`; hay que ser menos exigente con la justificación de párrafos cortos.)

11.3 Cómo guardar y copiar cajas

Una caja cualquiera (`\hbox`, `\vbox`, `\vtop` o `\vcenter`) puede guardarse en un registro de caja, con el comando `\setbox`. Si se quiere reservar la caja por un tiempo, es mejor guardarla en un registro designado con `\newbox`; pero si se pretende usarla en seguida, es mejor usar los registros de caja $0, 1, \ldots, 9$, que se mantienen disponibles para uso temporal. Se puede entonces guardar cajas con:

```
\setbox0 = \hbox{...}
\setbox1 = \vbox{...}
\setbox4 = \vtop{...}
\setbox7 = \hbox{$\vcenter{...}$}
```

La última asignación es un tanto complicada, porque `\vcenter` solo opera en el modo matemático; un `$` debe ocurrir en el modo horizontal, así que el `$\vcenter{...}$` debe empacarse en un `\hbox`, que luego se guarda en el registro `\box7`.

Los comandos `\box` n y `\copy` n escriben el contenido del registro de caja n. El comando `\box` *deja vacío el registro* después de extraer su contenido; `\copy` extrae el contenido sin cambiar el registro, es decir, escribe una copia de su contenido. Si este registro ha sido designado mediante `\newbox\micaja`, y luego se asigna `\setbox\micaja=\hbox{aquella}`, entonces `\copy\micaja` o `\box\micaja` imprimirá la palabra 'aquella' en la fuente que estaba en uso a la hora de empacarla con `\setbox`.

11.4 Uso de resortes en cajas

Si el contenido de una caja pedida con `to` (o con `spread`) mide menos que las dimensiones establecidas de la caja, habrá un "underfull box" si no se justifica el contenido con resortes. Compare

`\hbox{A\hfil B\hfil C}`	ABC
`\hbox to 10pc{A \hfil B \hfil C}`	A B C
`\hbox to 10pc{A \dotfill\ B \dotfill\ C}`	A ..... B C
`\hbox to 10pc{A \hrulefill\ B \hrulefill\ C}`	A _____ B _____ C

En el primer caso, se usa la longitud natural del contenido; como `\hfil` es un sinónimo de `\hskip 0pt plus 1fil`, su longitud natural es 0 pt y no se nota su presencia. En la segunda caja, se usa la estirabilidad de los resortes `\hfil` para justificar el ancho de la caja a 10 pc.

Los comandos `\dotfill` y `\hrulefill` son *resortes visibles*; producen una línea de puntos y una regla horizontal cuya "longitud natural" es de 0 pt (por lo tanto, invisible) pero que pueden estirarse como `\hfil` para rellenar un espacio disponible. Hay otros dos resortes visibles, llamados `\rightarrowfill` y `\leftarrowfill`, que producen flechas como `\to` y `\gets` (pero en modo horizontal) cuyas aristas se estiran como resortes.

Hay que tener cierto cuidado en usar resortes fuera de cajas, porque TEX desecha goma y resortes en algunas ocasiones cuando está confeccionando la lista vertical principal para la página actual. En particular, TEX desecha goma (y resortes) y penalties al inicio de una página, para que todas las páginas empiecen en una posición uniforme. Así, el código

```
\def\newpage{\vfill\eject}
⟨texto⟩
\newpage\newpage
⟨texto⟩
```

no genera una página en blanco entre los dos bloques de texto. El primer `\newpage` corta la página. La siguiente página empieza con el resorte `\vfill`, que se desecha, y el `\eject`, cuya expansión es `\par\penalty-10000` (*TEXbook*, p. 353). Un `\par` es ignorado en el modo vertical, y el penalty al inicio de la hoja también queda desechado. En resumen, el segundo `\newpage` no tiene efecto. Para obtener una página en blanco, hay que teclear

```
\newpage\hbox{}\newpage
```

ya que la caja horizontal, aunque sea vacía, convierte el modo de vertical en horizontal y el segundo `\newpage` tiene su efecto esperado. Para hacer más cómodo el trabajo, Plain TEX define el macro `\null` como sinónimo de `\hbox{}` para situaciones como ésta.

En caso de duda, se puede prevenir la pérdida de resortes en esta forma al delimitarlos por `\null`. Por ejemplo, el párrafo

```
\null\hfill Al centro! \hfill\null
```

centra una línea sin necesidad de usar una caja. El código

```
\line{\hfill Al centro! \hfill}
```

alcanza el mismo objetivo con una caja larga, usando resortes para rellenar su contenido.

11.4.1. Traslapos de texto.

Para ilustrar las posibilidades de obtener efectos tipográficos con cajas y goma, consideremos los macros `\rlap` y `\llap` de Plain TEX, que colocan texto a la derecha [respectivamente, a la izquierda] de la posición actual sin avanzar. Con `\rlap{`⟨*algo*⟩`}` se puede colocar ⟨*algo*⟩ hasta en el margen derecho; `\llap{`⟨*algo*⟩`}` lo coloca a la izquierda, quizás en el margen izquierdo, o en la caja de sangría al inicio del párrafo. Un carácter sobreescrito como ∉ puede obtenerse con `\rlap/c` o

bien con `/\llap{c}` (`\rlap/` escribe una / y retrocede para poder colocar la 'c'; `\llap{c}` retrocede por la anchura de 'c' y luego escribe esta letra, en cima de la / ya puesto).

La definición `\rlap{`⟨*algo*⟩`}` es `\hbox to0pt{`⟨*algo*⟩`\hss}`. Esta caja vierte su contenido a la derecha de su posición actual y el punto de impresión no avanza, ya que la caja tiene una anchura de 0 pt. La goma `\hss` se contrae por la anchura natural de ⟨*algo*⟩, y esto evita una protesta sobre un "`Overfull \hbox`": recuerde que TEX permite que el contenido de una caja salga del recinto de la caja misma. Por lo tanto, el `\hss` funciona en la práctica como un *retroceso no destructivo* que no borra el ⟨*algo*⟩ impreso.

11.5 Manipulación de cajas

Una vez que una caja haya recibido su contenido, es posible cambiar el tamaño *aparente* de la caja: es cuestión de cambiar los parámetros `\ht`, `\dp`, `\wd` de esta caja. Estos cambios no desplazan el punto de referencia. Luego, el contenido se imprimirá igualmente, pero TEX será "engañado" cuando hace sus cálculos para las posiciones de las cajas vecinas. Ejemplo: 0

```
\setbox0 = \hbox{payaso}
\dp0 = 0pt
$\underline{\box0}$
```

payaso

Este código engaña a TEX para que "no vea" las partes de las letras p, y que desciendan por debajo de la línea de base; luego la barra queda sobreescrita a estas letras.

⟦Si se requiere desplazar el punto de referencia, se disponen de `\raise`⟨*dimensión*⟩⟨*caja*⟩ y `\lower`⟨*dimensión*⟩⟨*caja*⟩ en el modo horizontal; análogamente, hay `\moveright`⟨*dimensión*⟩⟨*caja*⟩ y `\moveleft`⟨*dimensión*⟩⟨*caja*⟩ que funcionan en el modo vertical. Por ejemplo,

```
\def\Mc{M\raise.5ex\hbox{c}}
```

es un macro útil para bibliografías que citan libros publicados por M$^{\rm c}$Graw-Hill.⟧

El macro de manipulación más útil que tiene Plain TEX es `\smash`, que en el modo horizontal hace lo siguiente:

```
\setbox 0 =\hbox{#1}
\ht0 = 0pt  \dp0 = 0pt  \box0
```

que agarra el argumento de `\smash`, lo pone momentáneamente en una caja, aplasta la extensión vertical de la caja y luego suelta el contenido de la caja aplastada. El macro `\smash` también funciona en el modo matemático. Por ejemplo:

`$\smash{\pmatrix{a & b \cr c & d \cr}}$.` $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

También hay un macro `\phantom`, que imprime una caja en blanco del tamaño de su contenido. Funciona en los modos horizontal y matemático. Por ejemplo:

`AB\phantom{CD}E`	AB E
`$x - 2y + \phantom{3}z.$`	$x - 2y + \ z.$

12 Macros avanzados

12.1 Definición con `\def` y `\let`

Los macros se definen normalmente con `\def\nuevomacro{...}`. Con este código, TEX crea una nueva ficha `\nuevomacro` y la reconoce como sinónimo de su "texto sustituto". Si este texto cambia de sentido, el significado de `\nuevomacro` cambiará también. Ejemplo:

```
\def\a{aaa}
\def\bb{\a\a}\bb b
\def\a{cc}\bb
```

imprimirá 'aaaaaab cccc' por las dos instancias del macro `\bb`. Así, el mismo macro puede tener variar expansiones distintas en diferentes oportunidades.

Esto puede resultar útil, pero también peligroso, por razones que deben de ser obvias. Para prevenir contra lapsos de memoria (humana), TEX proporciona un mecanismo de definir un macro que no está sujeto a cambios posteriores. El código

```
\let\algo = ⟨ficha⟩
```

asigna a la palabra de control `\algo` el significado de ⟨*ficha*⟩ (que puede ser un carácter u otro macro) *en el momento de la asignación*. Si la ⟨*ficha*⟩ es un macro cuyo sentido es alterado posteriormente con `\def`, la palabra de control `\algo` conserva el sentido original. Esto es útil para abreviaturas:

```
\let\del = \partial
\let\mete = \hookrightarrow
\let\to = \rightarrow
```

y más útil aún para conservar viejas formas de un macro:

```
\let\dotunder = \d
\def\d{\delta}
```

que es apropiado para manuscritos con muchos δ. Como `\d` ya significa un acento poco usado (punto debajo de la letra que sigue), se conserva la definición original en `\dotunder`.

El lado derecho de una asignación `\let` puede ser un *carácter*. En Plain TEX se establece =10pc

```
\let\bgroup={  \let\egroup=}
```

que definen `\bgroup` y `\egroup` cómo sinónimos de los caracteres que abren y cierran grupos. Por lo tanto, `\bgroup` y `\egroup` abren y cierran grupos también. Sin embargo, estos agrupadores no son reconocidos como tales dentro del texto sustituto de un macro. Luego, se puede definir

```
\def\abrircajita{\vbox to 0pt\bgroup\hsize=10pc}
\def\cerrarcajita{\vss\egroup}
```

que permite empacar unos párrafos delgados de texto (que, por tener una altura total de 0 pt, traslaparán lo que sigue) con `\abrircajita ... \cerrarcajita`. El texto intermedio no debe ser muy extenso, porque TEX no puede cortar una página hasta que la cajita esté cerrada con el `\egroup` final.

Para *quitar* la definición de un macro (como `\>`, quizás) sin redefinirlo, se puede teclear

```
\let\> = \undefined
```

ya que el macro `\undefined` no ha sido definido por Plain TEX. El uso posterior de `\>` causa una protesta sobre un `Undefined control sequence`, hasta que el macro `\>` sea redefinido explícitamente. (Esta anulación del macro `\>` está incorporado al formato $\mathcal{AMS}$-TEX.)

12.2 Macros locales y globales

Un macro definido con `\def` o `\let` es "local": si la definición ocurre dentro de un grupo, esta definición desaparece al final del grupo. Por ejemplo, una fórmula larga con muchas δ y ϵ podría teclearse con abreviaturas *temporales*:

```
$$
\def\d{\delta} \def\e{\epsilon}
⟨fórmula⟩
$$
```

usando `\d`, `\e` dentro de la ⟨*fórmula*⟩. Los `$$` encierran un grupo; al terminar la fórmula, `\d` recupera su definición original (un acento) y `\e` pierde la suya, pues antes de la fórmula no estaba definido.

Para conservar una definición hecha dentro de un grupo, hay que establecerla con `\global\def` o `\global\let`; TEX tiene un sinónimo corto `\gdef` para `\global\def`. Veamos la definición de `\obeylines` (*TEXbook*, p. 352):

```
{\catcode`\^^M=\active%
 \gdef\obeylines{\catcode`\^^M=\active \let^^M=\par}%
 \global\let^^M=\par}
```

Este código hace lo siguiente. El símbolo `^^M` es un convenio para denotar el carácter 13 de ASCII (vea el *TEXbook*, p. 47), que es el retorno de carro ↵; el código `\catcode`\⟨carácter⟩=\active` cambia la categoría de cualquier carácter a "activo" (categoría 13); `^^M` funciona como macro *dentro del grupo* encerrado por las llaves inicial y final, y es por eso que se colocan los `%` en las líneas internas para suprimir los ↵ a su derecha. El `\global\let` define `^^M` como sinónimo de `\par`, toda vez que el retorno ↵ sea activo (normalmente es inactivo, con categoría 5); y `\obeylines` provee un macro que establece estos convenios en forma temporal. Como `\obeylines` recibe su definición dentro de un grupo, se usa `\gdef` para que esta definición no desaparezca en seguida.

Cualquier asignación de parámetro dentro de un grupo es también local; pero si la asignación es precedida por `\global`, persiste fuera del grupo ambiente. Veamos la definición de `\advancepageno` (*TEXbook*, p. 362):

```
\def\advancepageno{\ifnum\pageno<0 \global\advance\pageno by -1
                       \else \global\advance\pageno by 1 \fi}
```

que incrementa el registro `\pageno` por ± 1.

12.3 Macros con contadores

Los macros "sofisticados" que más se emplean son aquellos que usan alguna forma de numeración automática. Por ejemplo, si se quiere colocar las notas al pie en orden numérico consecutivo, se necesita un registro `\count` para llevar la cuenta de las notas ya puestas. Recordamos que `\footnote` requiere dos argumentos, la marca y el texto de la nota. Se podría teclear:

```
\newcount\alpieno  \alpieno = 0
\def\notaalpie{\advance\alpieno by 1
 \footnote{$^{\the\alpieno}$}}
```

para establecer `\notaalpie` como un macro (con un solo argumento, el texto de la nota) que coloca el número en forma automática. La primera nota será numerado 1, la siguiente 2, y así sucesivamente.

El formato LATEX usa contadores para numeración de secciones, subsecciones, ecuaciones, notas al pie, listas itemizadas, etcétera. En su ambiente* de listas `enumerate`, el macro `\item` (que es diferente del `\item` de Plain TEX) usa un contador `enumi`. Podemos emular su funcionamiento así:

```
\newcount\enumerador
\def\primeritem{\enumerador=1 \item{1.}}
\def\otroitem{\advance\enumerador by 1
              \smallskip \item{\the\enumerador.}}
```

y entonces podemos hacer una lista de ítemes con `\primeritem`, `\otroitem`, `\otroitem`, ... que serán numeradas 1., 2., 3., ..., y serán separadas por un `\smallskip` entre cada par de ítemes.

* En la § 13.3 hablaremos de los "ambientes" de LATEX.

12.4 Expansión de macros

Cuando TEX lee un macro, lo que hace normalmente es reemplazarlo por su "texto sustituto"; si este texto contiene macros, serán leídos y reemplazados por sus propios textos sustitutos, etc. Este proceso es lo que llamamos "expansión del macro". A veces la expansión es inapropiada: por ejemplo, en `\def\nuevomacro{...}`, no se expande `\nuevomacro` de una vez, porque aún no tiene texto sustituto hasta que TEX haya leído el grupo `{...}`. Por ende, se suprime la expansión de macros inmediatamente después de `\def` (y después de `\let` y `\gdef` y `\font`). También se suprime la expansión cuando TEX está leyendo argumentos de un macro, el texto sustituto de una definición, el preámbulo de un `\halign` y en algunas otras ocasiones (para la lista completa, vea el *TEXbook*, p. 215).

Es posible *suprimir la expansión* de un macro `\delicado` al precedirlo con `\noexpand`: TEX convierte `\noexpand\delicado` en la ficha '`\delicado`' sin más expansión.

Por otro lado, a veces se requiere reactivar una expansión normalmente suprimida. Por ejemplo, una lista de fichas en `\uppercase{`⟨*fichas*⟩`}` no se expande a la hora de su lectura inicial. Entonces

```
\def\clave{Alguna Cosa} \uppercase{\clave}
```
Alguna Cosa

no cambia los caracteres a mayúsculas, pues `\uppercase` sólo afecta caracteres pero ignora macros, y no expande el macro `\clave`. Si hubiera un comando `\expand` para hacer una expansión inmediata de lo que sigue, tampoco serviría, pues en `\uppercase{\expand\clave}` los dos macros `\expand\clave` serían engullidas por `\uppercase` sin expansión. Pero TEX sí tiene un comando `\expandafter`, que hace lo siguiente:

`\expandafter`⟨*ficha*$_1$⟩⟨*ficha*$_2$⟩

expande la ⟨*ficha*$_2$⟩ (y otras fichas, si este posee un argumento) y luego coloca ⟨*ficha*$_1$⟩ *antes* de la expansión. Así, podemos decir

```
\def\CLAVE{\expandafter\uppercase{\clave}}
```

para obtener los caracteres de `\clave` en mayúsculas. Obsérvese que la segunda ficha después de `\expandafter` es el abre-grupo `{` que es incompleto en sí; luego se expande el texto equilibrado `{\clave}` en `{Alguna Cosa}`, y el texto sustituto de `\CLAVE` deviene en `\uppercase{Alguna Cosa}`, que produce ALGUNA COSA.

12.5 Comunicación con archivos externos

12.5.1. Comunicación con la terminal.

El comando `\message{`⟨*fichas*⟩`}` imprime un mensaje en el archivo TeX log: la lista ⟨*fichas*⟩ es expandida en caracteres que se imprimen luego en la terminal durante el proceso de levantamiento de texto. Por ejemplo:

```
\def\ecnum{4.6}
\message{Ojo: Revise la ecuacion \ecnum!}
```

colocado después de una ecuación de dudosa validez, produce los caracteres '`Ojo: Revise la ecuacion 4.6!`' en el archivo TeX log.

Un archivo fuente puede pedir una línea de texto de la terminal, que luego se convertirá en el texto sustituto de un macro. Por ejemplo,

```
\read16 to \nombre
```

es equivalente a `\def\nombre{`⟨*línea externa*⟩`}`, donde ⟨*línea externa*⟩ es lo que se teclea en la terminal. El levantamiento se detiene y se escribe '`\nombre=`' en la terminal para que el usuario teclee `Hermenogildes Garaicoechea Bunyakovskiy` (o lo que corresponde) seguido por un ↵ que termina la línea y devuelve el control al programa.

12.5.2. Lectura de líneas de archivos externos.

El comando `\read` también puede usarse con archivos externos (aparte de los que se leen con `\input`), hasta un máximo de 16 archivos, numerados de 0 a 15. (Por convenio, 16 significa la terminal.) Primero se abre un canal de lectura con

`\openin`⟨*entero*⟩ `=` ⟨*archivo*⟩

donde ⟨*entero*⟩ $\in \{0, 1, \ldots, 15\}$ y ⟨*archivo*⟩ es el nombre de un archivo en disco. (Se puede omitir la extensión '`.tex`' si el nombre del archivo termina con esta extensión.) El número ⟨*entero*⟩ puede ser designado por el comando `\newread` de Plain TEX, que es análogo a `\newcount` y reparte canales de lectura.

Por ejemplo, una lista de nombres (uno por línea del archivo) confeccionada con una base de datos en el archivo `nomlista.tex` puede ser leído y procesado por:

```
\newread\losnombres
\openin\losnombres = nomlista
\ifeof\else\read\losnombres to\nombre \procesar\nombre\fi
...
\closein\losnombres
```

La condicional `\ifeof` es "verdadera" cuando no quedan más líneas en el archivo externo. El comando `\closein` cierra el canal de lectura abierto con `\openin`.

12.5.3. Escritura de líneas en archivos externos.

La pareja natural de `\read` es `\write`, que escribe cosas en un archivo externo. Hay 16 canales de estos archivos (numerados de 0 a 15), abiertos con `\openout`, cerrados con `\closeout` y designados con `\newwrite`. Lo que se "escribe" en uno de estos canales es una lista de fichas:

`\write`⟨*entero*⟩`{`⟨*fichas*⟩`}`

enviará la lista de fichas (con sus macros expandidos) al archivo correspondiente. Si ⟨*entero*⟩ es 16, estas fichas serán enviadas a la terminal.

El formato LATEX mantiene un sistema de referencias e índices mediante un conjunto de archivos auxiliares. Podemos emular este sistema con el código:

```
\newwrite\ayuda
\openout\ayuda = \jobname.aux
```

El comando `\jobname` es el nombre del archivo que se está levantando (sin extensión '`.tex`'); si este archivo es `asco` o bien `asco.tex`, el canal `\ayuda` conducirá a un nuevo archivo `asco.aux`.

Podemos dividir el archivo fuente en secciones, enumeradas con un contador `\secno`, con un macro como el siguiente:

```
\def\seccion#1\par{\advance\secno by 1
        \bigbreak\noindent{\bf\the\secno.\enskip #1}%
        \write\ayuda{Parte \the\secno: hoja \the\pageno}%
        \par\nobreak\medskip}
```

el cual, entre otras cosas, escribe algo como '`Parte 3: hoja 17`' en el archivo `asco.aux` (si la tercera sección empieza en la página 17).

⟦ Para controlar la numeración de páginas en `\write\ayuda{...\the\pageno...}`, la expansión de la lista de fichas en un `\write` queda diferido hasta el momento de la ejecución del `\shipout` correspondiente en la rutina de salida. Si se desea una expansión inmediata de estas fichas, se puede decir `\immediate\write`. El comando `\typeout` de LATEX usa un `\immediate\write16` para enviar

líneas a la terminal; en contraste con `\message`, un `\write16` siempre comienza una línea nueva en el archivo TeX log.]]

Si se requiere *escribir un macro* en un archivo auxiliar con `\write`, hay que inhibir su expansión en el argumento de `\write`. Por ejemplo, podemos modificar la tercera línea de la definición de `\seccion` así:

```
\write\ayuda{\noexpand\secdeindice\the\secno:\the\pageno;}%
```

y ahora se enviará algo como '`\secdeindice 3:17;`' al archivo `asco.aux`. Ahora se puede formatear el índice apropiadamente con `\def\secdeindice#1:#2;{...}`.

12.5.4. Comunicación con la rutina de salida.

Un mecanismo análogo puede usarse para pasar listas de fichas a la rutina de salida. Por ejemplo, si el encabezado de la página depende del título de la sección actual, se requiere establecer una lista de fichas (el título mismo o una versión corta de él) que luego se incluirán en el `\headline`. Esto se hace mediante un sistema de "marcas" en el texto, establecidas con el comando `\mark`. Si el archivo fuente contiene

`\mark{`⟨*fichas*⟩`}`

TEX coloca una "marca" en la lista vertical principal en el sitio donde encontró el comando `\mark`; no se imprime ⟨*fichas*⟩ en este lugar sino que la marca los expande y mantiene en reserva la lista expandida de fichas.

Se obtiene acceso al contenido de una marca mediante los comandos `\botmark`, `\firstmark` y `\topmark`, que denotan la última marca en la página actual, la primera marca en la página actual y la última marca en la página anterior, respectivamente. Si no hay marca alguna en la página actual, estos tres comandos usan el `\botmark` anterior; inicialmente son listas de fichas vacías. Luego, el código

```
\headline={\tenrm\hfil\firstmark\hfil\folio}
\def\seccion#1\par{...\mark{#1}...}
```

colocará el título de la sección en el encabezado de la página donde comienza la sección, y en las páginas siguientes hasta que se inicie una nueva sección o se coloca una nueva marca en el archivo fuente.

12.6 Incorporación de archivos gráficos

La incorporación de dibujos y diversos elementos gráficos ajenos al texto estructurado (por ejemplo, fotografías, diagramas complejas, logos comerciales) es un tema de gran complejidad, tanto por la cantidad de formatos gráficos disponibles, como por las diversas maneras de incluirlos en un documento. Se trata de un problema cuya solución depende en alto grado del sistema operativo subyacente. Aunque algunos formatos ya se han estandarizado, como el Graphic Interchange Format (archivos `.gif`) o bien el Encapsulated PostScript (archivos `.eps`), todavía no hay un acuerdo universal acerca de su tratamiento en las diversas implementaciones de TEX.

TEX posee un mecanismo para agregar habilidades extras a esas implementaciones, mediante el comando `\special` (*TEXbook*, p. 228). Un

`\special{`⟨*fichas*⟩`}`

crea un "quéeseso" ("whatsit"), sobre el cual TEX sabe únicamente su posición de referencia en la página. Para incluir un archivo de PostScript, por ejemplo, el programa reconoce una instruccion como `\special{ps:` ⟨*archivo*⟩`}` o quizás `\special{postscript` ⟨*archivo*⟩`}`, donde ⟨*archivo*⟩ es el nombre de un archivo `.ps` o bien `.eps`, con código PostScript.

No hay garantía alguna de que el efecto del `\special` sea la misma en dos sistemas diferentes; en ese sentido, el uso de este recurso arriesga perder la característica sobresaliente del TEX: su

portabilidad. Para minimizar ese riesgo, Tom Rocicki, el creador del interfase `dvips`, ha escrito un archivo auxiliar `epsf.tex` (con una versión `epsf.sty` para LaTeX), que proporciona una manera estándar de incorporar archivos `.eps` en un documento levantado con TeX. Este archivo es muy recomendable para los que quieren incorporar gráficos hechas con PostScript en documentos hechas en Plain TeX.

Otra clase de extensión al TeX es la que permite usar texto en colores (o en escalas de gris). Esto requiere una implementación que incluye un comando `\special{color` ⟨*tinta*⟩`}`, donde ⟨*tinta*⟩ puede ser un color primario: `red`, `green` o `blue`, o bien alguna combinación de ellos. Todavía no hay acuerdo universal sobre los convenios que deben usarse, ya que el uso de colores depende fuertemente de las tecnologías de impresión, aun en plano desarrollo.

13 Archivos de Formato

Todos los macros de Plain TeX están definidos en el Apéndice B del *TeXbook*. Por ejemplo, `\centerline` y `\line` son macros de Plain TeX. Los ingredientes de sus expansiones finales, como `\hbox`, `\hsize` y `\hss`, no son macros sino que son comandos "primitivos" de TeX. Hay 325 comandos primitivos inede TeX y 554 macros de Plain TeX en total: todos están en el índice del *TeXbook*. (Algunas implementaciones, como *Textures*, tienen el formato Plain TeX incorporado; pero todas poseen un preformato, llamado IniTeX o VirTeX, que incluye solamente los comandos primitivos, para poder instalar otros formatos.) El formato alternativo de mayor uso es LaTeX, un formateador de documentos; también hay un formato $\mathcal{AMS}$-TeX, patrocinado por la AMS (American Mathematical Society); y una versión extendida de Plain TeX llamado EPlain, de Karl Berry, entre otros. Hay varios formatos que extienden el Plain TeX o el LaTeX para permitir la inclusión de dibujos con comandos de TeX (sin recurrir a interfases con archivos externos); entre ellos, se destacan el PiCTeX de Michael Wichura, el DraTeX de Eitan Gurari, y especialmente el XY-pic de Kristoffer Rose y Ross Moore.

El Apéndice E del *TeXbook* incluye muestras de colecciones de macros para diversos propósitos: cartas, afiches musicales y el propio *TeXbook*. Estos archivos de macros pueden convertirse en una forma compacta, llamado *formato*, para uso eficiente con TeX. Se prepara un archivo de macros, y en vez de terminarla con `\bye` o `\end`, se coloca `\dump` como última instrucción. El resultado de levantar este archivo es un archivo binario que TeX puede leer a alta velocidad, para mayor rapidez en el levantamiento de otros documentos con este formato. En particular, LaTeX requiere este preprocesamiento, para poder interpretar el comando `\documentclass` que aparece al inicio de cada uno de sus documentos.

13.1 Introducción a Plain TeX

El programa TeX funciona en dos niveles. Sus 325 comandos *primitivos* hacen el trabajo esencial de levantamiento de texto. Luego hay que establecer un juego de *macros* que combinan los comandos primitivos en una forma inteligible para el usuario. (Los capítulos 24 a 26 del *TeXbook* tienen una descripción sintáctica exacta de los comandos primitivos, para que el usuario pueda salir de dudas.) Por ejemplo, `\end` es un comando primitivo ("termine el trabajo") pero `\bye` es un macro definido en Plain TeX como sinónimo de `\par\vfill\par\penalty-20000\end` (cinco comandos primitivos). Echaremos ahora un vistazo a la forma de montar los macros esenciales, a partir de los comandos "de bajo nivel" de TeX.

13.1.1. Códigos.

Primero se establecen una serie de códigos internos. Un comando

`\catcode‘\`⟨*carácter*⟩ `=` ⟨*entero*⟩

asigna al ⟨*carácter*⟩ la categoría ⟨*entero*⟩ (entre 0 y 15). Cuando TeX se inicializa por primera vez, las letras (del abecedario inglés) ya tienen categoría 11 y `\` tiene categoría 0. Luego se establece `\catcode‘\{=1`, etc., en conformidad con la tabla de § 6.1.

Se asignan luego otros códigos que identifican caracteres como símbolos matemáticos y como delimitadores en fórmulas. Por ejemplo, se incluye

```
\mathcode`\- = "2200
```

que indica que en el modo matemático, un guión debe interpretarse como el carácter `"00` (numeración hexadecimal) de la familia matemática `2` (una fuente de símbolos `cmsy`), el cual es el signo menos, y le asigna la clase `2` (una operación binaria). A los delimitadores (paréntesis, corchetes, etc.) que son caracteres del teclado, se asigna un `\delcode` con una codificación aún más compleja.

Algunos macros de Plain TEX deben ser inaccesibles al usuario para que no los modifique sin intención. Para hacer eso, se asigna ahora `\catcode`\@=11` y se reasigna `\catcode`\@=12` al final del archivo `plain.tex`. La idea es que el carácter `@` sea reconocido temporalmente como "letra" dentro del archivo del formato. Entonces `\m@th` es una palabra de control dentro de `plain.tex` pero en un documento normal se lee como un juego de 4 fichas `\m`, `@`, `t`, `h`. Si este comando fuera denominado `\math` o `\moth` o `\myth`, un usuario podría —con la mejor intención— redefinirlo con `\def`, produciendo un caos impresionante a la hora de levantar el documento.*

13.1.2. Designación de registros.

Plain TEX define repartidores de registros: `\newcount`, `\newdimen`, `\newskip`, `\newmuskip`, `\newbox`, `\newhelp` (para mensajes de ayuda a la terminal), `\newtoks`, `\newread`, `\newwrite`, `\newfam` (para familias de símbolos matemáticos), `\newinsert` (para inserciones flotantes) y finalmente `\newlanguage` (para patrones de división silábica).

⟦Estos macros se definen con `\outer\def`: si `\macromio` es definido con

```
\outer\def\macromio{...}
```

TEX protestará furiosamente si `\macromio` aparece en el texto sustituto o en el argumento de cualquier otro macro. Ciertos macros importantes deben ser "externos" en este sentido: entre ellos están `\+`, `\beginsection`, `\proclaim` y `\bye`.⟧

Luego Plain TEX define `\outer\def\newif#1{...}`, cuya definición usa muchos recursos de bajo nivel de TEX; basta decir que `#1` debe ser una palabra de control que empieza con los caracteres `\`, `i`, `f`, y que `\newif\ifsera` crea dos nuevos macros `\seratrue` y `\serafalse`, y también crea el macro `\ifsera` como una condicional que reconoce a éstos como sus valores "verdadero" y "falso".

13.1.3. Parámetros.

Plain TEX ahora asigna valores a los parámetros primitivos de TEX. Aparecen

```
\hsize=6.5in  \vsize=8.9in  \parindent=20pt  \overfullrule=5pt
```

y muchos otros. Cabe mencionar

```
\pretolerance=100  \tolerance=200
```

que establecen límites estrictos a la estirabilidad de renglones en la justificación de un párrafo de texto. TEX intenta dividir el párrafo en renglones con espaciamiento uniforme y sin guiones, aceptando un monto de `\pretolerance` de "fealdad" o "no uniformidad" del espaciamiento; si esto no resulta posible, TEX hace un segundo intento permitiendo guiones y aceptando una "fealdad" de `\tolerance`. (Si esto también falla, habrá un "overfull hbox".) En párrafos delgados con un menor `\hsize`, se recomienda aumentar `\pretolerance` y `\tolerance` proporcionalmente.

Otros parámetros no primitivos se definen ahora; entre ellos:

```
\newskip\smallskipamount  \smallskipamount = 3pt plus1pt minus1pt
\newdimen\jot             \jot = 3pt
```

(un `\jot` es una unidad para estirar el espacio entre renglones de un despliegue multilineal).

* Para enterarse del significado de `\m@th`, vea el *TEXbook*, p. 353.

13.1.4. Fuentes.

El próximo paso es cargar las fuentes. Con `\font\tenrm = cmr10` se lee los parámetros métricos de la fuente `cmr10` y se le designa con la palabra de control `\tenrm`. Plain TEX usa 16 fuentes de la familia Computer Modern:

Nombre	*Fuente*	*Nombre*	*Fuente*
`\tenrm`	`cmr10`	`\teni`	`cmmi10`
`\sevenrm`	`cmr7`	`\seveni`	`cmmi7`
`\fiverm`	`cmr5`	`\fivei`	`cmmi5`
`\tenbf`	`cmbx10`	`\tensy`	`cmsy10`
`\sevenbf`	`cmbx7`	`\sevensy`	`cmsy7`
`\fivebf`	`cmbx5`	`\fivesy`	`cmsy5`
`\tensl`	`cmsl10`	`\tenex`	`cmex10`
`\tenit`	`cmti10`	`\tentt`	`cmtt10`

Figura 13.1. Las fuentes usadas por Plain TEX

En seguida se establecen ciertos convenios para el uso de estas fuentes en el modo matemático. El comando `\tenrm` solo tiene efecto en el modo horizontal. El modo matemático emplea sus fuentes a través de "familias": `\fam0` para la fuente de texto (aquí `cmr`), `\fam1` para la fuente matitálica `cmmi`, `\fam2` para la fuente de símbolos `cmsy`, `\fam3` para la fuente de símbolos extensibles `cmex`; hay 12 familias más para otras fuentes (que pueden repartirse con `\newfam`). Encontramos ahora el código:

```
\textfont0=\tenrm  \scriptfont0=\sevenrm  \scriptscriptfont0=\fiverm
\def\rm{\fam0 \tenrm}
```

que hace que `\rm` signifique `\tenrm` en el modo horizontal, pero `\fam0` en el modo matemático. La familia 0 usa `\tenrm` en los tamaños `\textstyle` y `\displaystyle`, usa `\sevenrm` en el tamaño `\scriptstyle` y usa `\fiverm` en el tamaño `\scriptscriptstyle`. Por ende, se puede usar `\rm` por doquier y TEX se encargará de interpretarlo.

Se establecen convenios análogos para que `\it`, `\sl`, `\bf`, `\tt` puedan usarse dentro y fuera de modo matemático. También se definen

```
\def\mit{\fam1}   \def\cal{\fam2}
```

que permite el uso de los cambios de fuente `\mit` y `\cal` en el modo matemático solamente.

13.1.5. Macros para texto.

Sigue una larga lista de macros para formatear párrafos de texto. Aquí se definen `\obeylines`; `\quad` y `\qquad`; `\smallskip`, `\medskip`, `\bigskip`; `\break`, `\nobreak`, `\eject`; `\line`, `\leftline`, `\rightline` y `\centerline`; `\strut`; `\item`, `\itemitem` y `\narrower`; `\proclaim` (para formatear enunciados de teoremas) y `\beginsection` (para formatear títulos de secciones); `\raggedright`; `\dots`; acentos de texto; resortes visibles como `\dotfill` y `\hrulefill`; y `\bye`. También se establecen aquí los macros para tabulación `\+`, `\settabs` y `\cleartabs`, que usan una compleja manipulación de cajas vacías para establecer las posiciones de los tabuladores.

13.1.6. Macros para matemáticas.

La parte más larga de `plain.tex` se dedica a macros para fórmulas matemáticas. La mayoría de los símbolos se declaran con `\mathchardef`. Por ejemplo, `\mathchardef\alpha="010B` asigna el nombre `\alpha` a la posición `"0B` de la familia `1`, que es el símbolo α, con clase `0` (símbolo ordinario). Los delimitadores `\langle`, etc., son declarados de modo similar com `\delimiter`. Los acentos matemáticos `\acute`, etc., se definen con el comando primitivo `\mathaccent` para distinguirlos de los acentos de texto, que usan `\accent`.

Las flechas largas como `\longrightarrow` $\longrightarrow$ y `\Longrightarrow` $\Longrightarrow$ se forman de combinaciones $-\!\!\longrightarrow$ y $=\!\!\Rightarrow$ con un retroceso intermedio (`\mkern-3mu`) para eliminar la separación. En la confección de los tipos Computer Modern, se previó que los caracteres $-$ y $=$ estuvieran a la altura exacta para pegarse justamente con las barras horizontales de las flechas. Las definiciones precisa de $\longrightarrow$ y $\Longrightarrow$ son:

```
\def\relbar{\mathrel{\smash-}}  \def\Relbar{\mathrel=}
\def\joinrel{\mathrel{\mkern-3mu}}
\def\longrightarrow{\relbar\joinrel\rightarrow}
\def\Longrightarrow{\Relbar\joinrel\Rightarrow}
```

y el usuario podría definir flechas aun más largas con:

```
\def\Verylongrightarrow{\Relbar\joinrel\Longrightarrow}
```
$\Longrightarrow$

si tuviera la necesidad de emplearlas.

Algunos operadores son declarados usando `\mathop`. Tenemos, por ejemplo,

```
\def\cos{\mathop{\rm cos}\nolimits}
\def\det{\mathop{\rm det}}
```

que define `\cos` y `\det` como palabras con letra romana con un espaciamiento alrededor que es apropiado para operadores grandes. El `\nolimits` indica que los subíndices y exponentes se colocan a la derecha — como en $\cos^2\theta$ — y no arriba y abajo, en fórmulas desplegadas. El mismo mecanismo puede emplearse para definir `\sen`, `\sech`, `\tg` y otros "operadores de palabra" que Plain TEX omite.

Vale la pena mencionar algunos macros que dan efectos especiales. Aquí se definen `\smash`, que imprime su argumento, pero esconde su extensión vertical; y `\phantom`, que mantiene las dimensiones de su argumento pero no lo imprime. Tanto `\smash` como `\phantom` funcionan en modo horizontal y en modo matemático. Además, hay un macro `\buildrel` que construya relaciones compuestas (en modo matemático):

```
$\buildrel\alpha\over\longrightarrow$
```
$\buildrel\alpha\over\longrightarrow$

```
$\buildrel\rm def\over=$
```
$\buildrel\rm def\over=$

En general, `$\buildrel #1\over #2$` coloca `#1` en cima de `#2` y, usando `\mathrel{...}`, declara que es resultado es una relación.

Plain TEX provee dos clases de alineamientos matemáticos: matrices como `\matrix` y `\pmatrix` y fórmulas con `\eqalign` se montan con un `\halign` dentro de un `\vcenter`. Para otras fórmulas multilineales, se definen `\eqalignno`, `\leqalignno` y `\displaylines` como macros que emplean un `\halign to \displaywidth{...}`. Luego un despliegue multilineal puede tener varios `\eqalign` pero solo uno de cualquiera de los otros.

13.1.7. Macros para la rutina de salida.

Aquí Plain TEX define los macros `\folio`, `\nopagenumbers`, etc., ya discutidas en la § 9.1. Un macro complejo es `\footnote`, que usa una inserción flotante (designado con `\newinsert\footins`) para guardar el contenido de la nota. Con `\newinsert\topins` se abren registros para guardar otras inserciones flotantes creados con los macros `\topinsert`, `\midinsert` o `\pageinsert`. Estos abren una caja vertical con `\vbox\bgroup`, que debe cerrarse con `\endinsert` (cuya definición incluye el `\egroup` correspondiente). Los macros para `\plainoutput` discutidas en la § 9.1 reciben sus definiciones aquí. Luego Plain TEX establece su rutina de salida con `\output={\plainoutput}`.

13.1.8. Otros macros.

Ahora `plain.tex` contiene `\input hyphen`, para leer el archivo auxiliar `hyphen.tex` que contiene padrones para división silábica (en inglés).

Después, Plain TEX define `\magnification` y declara `\normalbaselines` (interlineación de 12 pt), `\rm` (letra romana) y `\nonfrenchspacing` (espaciamiento extra después de signos de puntuación) para inicializar el trabajo que sigue. Finalmente, esconde sus macros privados con la declaración `\catcode‘\@=12`.

13.2 Introducción al $\mathcal{AMS}$-TEX

$\mathcal{AMS}$-TEX es un formato para levantar artículos matemáticos en una forma que aproxima el estilo de una revista especializada. Está patrocinada por la AMS (American Mathematical Society); las revistas de la AMS y algunas otras aceptan manuscritos — de calidad y originalidad suficientes — hechos con $\mathcal{AMS}$-TEX.

$\mathcal{AMS}$-TEX no rehace el trabajo de Plain TEX, sino que *agrega varios macros más* a los que Plain TEX ya define. Los aspectos en donde extiende Plain TEX de modo significativo son: organización del documento, formateo de fórmulas, uso de fuentes especiales y chequeo de sintaxis. Alguién que ya conoce bien al Plain TEX será capaz de aprender $\mathcal{AMS}$-TEX en una tarde.

13.2.1. Organización del documento.

Se da por un hecho que el usuario de $\mathcal{AMS}$-TEX quiere escribir un artículo o algo muy parecido. Se preve un título destacado al inicio, una sección de referencias al final, y una organización del texto principal en secciones y subsecciones.

La parte titular se incluye entre los comandos `\topmatter` y `\endtopmatter`. Allí se colocan el título, el o los autores, su o sus direcciones, etc., dentro de bloques `\title...\endtitle`, `\author...\endauthor`, `\address...\endaddress`, etc. Un resumen del artículo va dentro de un bloque `\abstract...\endabstract`. El formato $\mathcal{AMS}$-TEX se encarga de eligir estilos, tamaños, justificaciones, etc., para estos bloques de texto.

Al final del artículo hay una sección `\Refs...\endRefs`, en donde aparecen referencias individuales en bloques `\ref...\endref`. La presentación de una lista de referencias debe seguir una normativa especial establecida por las revistas de matemáticas; $\mathcal{AMS}$-TEX se encarga de interpretar esta normativa y traducirla en un texto levantado: vea el Apéndice B del *Joy of TEX*.

La rutina de salida de $\mathcal{AMS}$-TEX coloca el contenido de los bloques `\title` y `\author` en los encabezados derecha e izquierda de las hojas del artículo. Los títulos de secciones deben incluirse en bloques como `\head...\endhead` y `\subhead...\endsubhead`.

Al inicio del documento, deben colocarse los macros específicos de ese trabajo, definidos con `\define\mimacro{...}`. (El `\define` es una versión de `\def`, disponible solamente en $\mathcal{AMS}$-TEX, con ciertas protecciones incorporadas para que no se cambie un macro ya definido: para poder cambiar un macro viejo, hay que teclear `\redefine\viejomacro{...}`.) Luego sigue el bloque `\topmatter...\endtopmatter`. Después viene el texto principal del documento, que debe empezar con `\document` y terminar con`\enddocument` (que es esencialmente un sinónimo de `\bye`).

13.2.2. Formateo de fórmulas.

Se levantan fórmulas como en Plain TEX. Lo que contribuya $\mathcal{AMS}$-TEX es una versión muy pulida del formateo de despliegues multilineales. Una matriz se levanta con:

Plain TEX:	`\pmatrix{...\cr...\cr...\cr}`
$\mathcal{AMS}$-TEX:	`\pmatrix...\\...\\...\endpmatrix`

y un alineamiento bilateral con:

Plain TEX:	`\eqalign{...&...\cr...&...\cr}`
$\mathcal{AMS}$-TEX:	`\aligned...&...\\...&...\endaligned`

En cada caso, $\mathcal{AMS}$-TEX mide la anchura del alineamiento total antes de colocar etiquetas a sus ecuaciones, y así previene contra traslapos de ecuación y etiqueta que a veces ocurren con Plain TEX. $\mathcal{AMS}$-TEX además posee una facilidad `\CD...\endCD` para formatear los temibles "diagramas conmutativos" de un modo "amistoso".

13.2.3. Fuentes de la AMS.

Otra particularidad de $\mathcal{AMS}$-TEX es el uso (opcional) de fuentes especiales de símbolos, hechos con METAFONT, a pedido de la AMS; también están disponibles, sin costo alguno, en formato

PostScript. Estos pertenecen a las series `msam` y `msbm` de la propia AMS, y `eufm` (Euler Fraktur Medium) del Proyecto Euler.† Estos fuentes se cargan con las instrucciones:

```
\loadmsam  \loadmsbm  \loadeufm
```

Enseguida se obtienen las siguientes familias de letras:

```
$\Bbb A \Bbb B ... \Bbb Z$          𝔸 𝔹 ... ℤ
$\frak A \frak B ... \frak Z$       𝔄 𝔅 ... 𝔷
$\frak a \frak b ... \frak z$       𝔞 𝔟 ... 𝔷
```

Los primeros son de "fuente abierta" o "blackboard boldface"; los otros son de estilo "fraktur", a veces mal llamada "letra gótica". ($\mathcal{AMS}$-TEX permite usar `\goth` como sinónimo de `\frak`, para no confundir éste con el `\frac` de fracciones.)

Se dispone también de dos fuentes de "negrilla matemática" en Computer Modern, `cmmib` y `cmbsy`. No son cargadas por Plain TEX, pero $\mathcal{AMS}$-TEX los carga con `\loadbold`. Es una peculiaridad de $\mathcal{AMS}$-TEX que no permite usar `\rm`, `\bf`, `\it`, `\sl`, `\tt` en el modo matemático (en eso difiere de Plain TEX), pero permite el acceso a *caracteres solitarios* en modo matemático con `$\roman x$`, `$\bold y$`, etc.

13.2.4. Chequeo de sintaxis.

$\mathcal{AMS}$-TEX incorpora una rutina para revisión sintáctica de un archivo fuente, sin levantamiento de páginas ni impresión de caracteres. Si al inicio de un archivo se coloca `\syntax`, se hace una rápida revisión del archivo y se generan los mensajes de error apropiadas en el TeX log; pero no hay aviso sobre "overfull boxes" ni hay emisión de páginas. Esto se logra al reemplazar todas las fuentes activas con una fuente especial `dummy` cuyos parámetros de tamaño son todos cero. Al no tener que calcular el espacio para las cajas de letra, el procesamiento es notablemente más rápido; una vez que se hayan eliminado los errores sintácticos, se puede suprimir el comando `\syntax` para ver el texto (con todos sus "overfull boxes").

13.2.5. Archivos de estilo.

El formato $\mathcal{AMS}$-TEX establece comandos genéricos; un estilo específico, para "preprints", está determinado por el archivo auxiliar `amsppt.sty`, que se carga con la instrucción

```
\documentstyle{amsppt}
```

al inicio del archivo. Otros estilos (para la producción final de revistas de la AMS) existen pero no han sido distribuidos al público. El archivo `amsppt.sty` establece el formateo exacto del "topmatter" y carga automáticamente las fuentes especiales de la AMS. También abre otro archivo auxiliar `amssym.tex` que contiene los `\mathchardef` de los símbolos nuevos de la AMS. Para el catálogo de estos símbolos, vea el *AMS Fonts User's Guide*. Este archivo puede abrirse también con el comando `\UseAMSsymbols`.

Para usar $\mathcal{AMS}$-TEX, debe haber en disco una copia del archivo de formato (hecho con `\input amstex` y `\dump`) en un directorio de formatos, y los archivos `amsppt.sty` y `amssym.tex` en un directorio de "inputs" (archivos auxiliares).

13.3 Introducción a LaTeX

El formato LaTeX tiene un propósito diametralmente opuesto al de $\mathcal{AMS}$-TEX: el de preparar TEX para documentos grandes de cualquier tipo, que no necesariamente contemplen el uso de muchas fórmulas matemáticas. El usuario de LaTeX puede emprender un libro, un informe largo, una tesis compleja, o una simple carta; pero será alguien que prefiere establecer la *estructura lógica* de su documento, sin preocuparse de los detalles de su *presentación visual*.

† Un proyecto tipográfico de la Universidad de Stanford, que diseñó los tipos para libros como *Concrete Mathematics* de Graham *et al.*

A modo de ejemplo, un libro se divide en capítulos, un capítulo en secciones, una sección en subsecciones, etc.; el autor debe poner el *título* de cada subdivisión, pero los detalles secretariales — numeración, referencias al índice, selección de estilos de letra, confección de encabezados, etc. — pueden dejarse al programa. LaTeX es ese secretario que enumera, organiza y formatea el documento (usando TeX) con la información mínima necesaria. Por ejemplo, el usuario puede teclear

```
\subsection{Las buenas nuevas}
```

en su archivo fuente `noticias.tex`, y LaTeX recordará que el capítulo actual es el tercero, la sección actual es la 5, que han habido 8 subsecciones anteriores en esta sección; luego colocará

3.5.9 Las buenas nuevas

en una línea aparte, con separación vertical apropiada antes y después, colocará una marca para pasar el título de la subsección a la rutina de salida y así modificar los encabezados de páginas, y escribirá información en un archivo `noticias.toc` para luego incluirla en el índice de contenidos.

En otras palabras, LaTeX asume la responsabilidad del formateo del documento, y permite que el autor se concentre en los detalles lógicos o "de alto nivel" de su documento. Las ventajas de este diseño son claras; pero hay una notable desventaja: la sintaxis adoptada por LaTeX para "esconder" las operaciones rutinarias de Plain TeX es tan compleja, o quizás más compleja aún, que la sintaxis de Plain TeX. Este desventaja está compensada por la gran cantidad de buenas manuales, para usuarios de todo tipo, que el mercado ofrece hoy en día. (En la bibliografía se mencionan algunas de estos manuales.)

La versión original de LaTeX, escrito entre 1985 y 1989 por Leslie Lamport, llegó hasta su versión 2.09. Desde 1994, ha sido reemplazado por una nueva versión estándar, a veces llamado LaTeX 2_ε (hay planes para desarrollar, en el futuro remoto, una versión definitiva que será el LaTeX 3). Aunque hay usuarios que todavía se obstinan en emplear la ya muy obsoleta LaTeX 2.09, la versión estándar es mucho más flexible.

Muchos otros formatos han sido modificados para hacerlos compatibles con LaTeX. En particular, debemos mencionar el $\mathcal{AMS}$-LaTeX, que es un paquete de archivos, confeccionados por la AMS, que agrega al LaTeX toda la funcionalidad de $\mathcal{AMS}$-TeX y que ya ha desplazado a éste en las preferencias de los autores.

Sin pretender, entonces, entrar en detalle respecto de las rutinas de LaTeX, pasaremos revista a sus rasgos más importantes de diseño documental.

13.3.1. Archivos de estilo.

Cada instalación de LaTeX coloca en un directorio apropiado, donde el programa TeX los puede leer, un gran número de archivos de estilo. Estos archivos, generalmente con una extensión `.cls`, determinan las características de la "clase de documentos" preferida. El más usado es la clase `article.cls`, para escribir artículos. También hay clases `report.cls`, `book.cls` y `letter.cls`, apropiados para informes más largos, libros de bolsillo y cartas, respectivamente. Para escribir un artículo en tamaño de letra de 12 pt, en papel europeo, hay que iniciar el documento con la instrucción

```
\documentclass[12pt,a4paper]{article}
```

que asigna los parámetros apropiados para ese documento, con las "opciones" (entre corchetes, separados por comas) elegidas. (Por defecto, un simple `\documentclass{article}` usará un tamaño de letra de 10 pt y papel americano (8.5 in $\times$ 11 in.)

Para elegir opciones suplementarias, disponibles en "paquetes" de archivos auxiliares, generalmente con una extensión `.sty`, la segunda instrucción debe ser `\usepackage{...}`. Por ejemplo,

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

prepara para el uso de las fuentes de la AMS (descritas en la sección 13.2), definiendo los nombres usuales de esos símbolos; además, define la sintaxis apropiada para los enunciados de teoremas y proposiciones al estilo de $\mathcal{AMS}$-LaTeX.

Hay una disposición para aquellas ocasiones, lamentablemente frecuentes, en donde uno recibe un documento levantado con la versión obsoleta de LaTeX. Estos archivos anacrónicos se detectan por la presencia de la instrucción `\documentstyle`, el antecesor de `\documentclass`. LaTeX puede procesar ese archivo, mediante su "modo de compatibilidad", pero con una notable lentitud. Es preferible reemplazar una instrucción como

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

por estas dos instrucciones:

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

que además lee el archivo `latexsym.sty` que define unos símbolos que formaban parte del viejo LaTeX (notablemente el cuadrado abierto `\Box`), pero que hoy en día son opcionales.

13.3.2. Archivos auxiliares.

Cada documento genera uno o más archivos '`.aux`' en donde se guarda información que registra contrarreferencias a otras partes del texto. Si se requiere un índice de contenidos, se crea un archivo '`.toc`' en donde los comandos de secciones escriben (con `\write`) información y macros para formatear dicho índice. Lo que hay que notar es que esta información no puede usarse hasta la *próxima* vez que se intenta levantar el documento. Luego, una de las primeras cosas que hace LaTeX es abrir y leer los archivos '`.aux`' y '`.toc`' generadas por "corridas" anteriores del mismo documento. Si han habido cambios en el archivo fuente, estos modificarán los archivos auxiliares y a veces se requieren varios ciclos para obtener un documento estable.

13.3.3. Ambientes.

El elemento de diseño más importante de LaTeX es el concepto de *ambiente*. Un ambiente es una parte del documento que debe formatearse de modo diferente del texto principal. Por ejemplo, una cita textual debe tener margenes encogidos; LaTeX establece un ambiente quote que coloca un brinco vertical antes y después de la cita, modifica `\leftskip` y `\rightskip` en forma simétrica, y quizás cambia el estilo de letra. El usuario debe teclear

```
\begin{quote}
⟨texto de la cita⟩
\end{quote}
```

y nada más; las instrucciones de formateo están metidos en los archivos '`.cls`' y por ende difieren de una `\documentclass` a otra.

Cada ambiente de LaTeX tiene un nombre ambiente (sin vara) y forma un bloque que debe empezar con `\begin{ambiente}` y terminar con `\end{ambiente}`. Hay ambientes para formatear texto (quote, quotation, verse, center, flushleft, flushright); para despliegues matemáticos (displaymath, equation, array, eqnarray); para figuras y tablas (figure, table); para referencias (thebibliography); para alineamientos (tabbing, tabular); para listas de ítemes (itemize, enumerate, description); en fin, para cualquier propósito que requiere un formateo fuera de lo común. (Hay un ambiente especial picture que tiene una limitada capacidad de graficación, usando algunas fuentes especiales de LaTeX; pero no es de fácil manejo.) Finalmente, hay una facilidad `\newenvironment` que permite al usuario definir nuevos ambientes o modificar ambientes existentes.

Diversos ambientes tienen una forma variante, indicado con un asterisco después del nombre. Por ejemplo, eqnarray genera una ecuación desplegada multilineal (parecida a `\eqalignno`) numerado automáticamente con un contador; su forma variante eqnarray* suprime el número de la ecuación.

Los ambientes para listas usan el comando `\item` para formatear las entradas de las listas. En el ambiente itemize, cada `\item` genera una seña flotante al inicio de un párrafo con sangría colgante (que podría ser •, −, ◇, o algo parecido). En el ambiente enumerate, el `\item` genera un número flotante a la izquierda; los ítemes consecutivos son numeradas 1, 2, 3, etc., en forma automática (usando un registro `\count`). Para una lista secundaria, es cuestión de encajar un ambiente dentro de otro:

```
\begin{itemize} ... \begin{itemize} ...
                    \end{itemize} ... \end{itemize}
```

13.3.4. Definición de macros en LaTeX.

El documento entero consta de dos partes: un "preámbulo" que puede tener definiciones de macros y asignaciones de parámetros, y el texto principal, que empieza con `\begin{document}`. Al final del archivo, se cierran las cuentas con `\end{document}`. El preámbulo empieza con el comando `\documentclass`, y termina con `\begin{document}`. Entre otras cosas, el `\begin{document}` lee cualesquiera archivos '`.aux`' que fueran generados por "corridas" anteriores del archivo fuente.

En el preámbulo, se pueden definir macros con `\newcommand`. Esto es análogo al `\define` de $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TeX, en cuanto averigua si el macro de marras ya está definido (en cuyo caso habrá que usar `\renewcommand`). Su manejo de parámetros difiere de Plain TeX; se escribe el número de parámetros, entre corchetes, antes del texto sustituto. Ejemplo: `\newcommand\feo[4]{...}` es equivalente a `\def\feo#1#2#3#4{...}` de Plain TeX.

Más generalmente, los macros de LaTeX permiten la sintaxis

`\lamacro[`⟨*opciones*⟩`]{`⟨*argumentos*⟩`}`

donde ⟨*opciones*⟩ son "argumentos opcionales", separados por comas; si no hay argumentos opcionales, los corchetes se omiten. Por ejemplo, `\newcommand` tiene un argumento opcional que debe ser un entero $n \in \{1, 2, \ldots, 9\}$, que indica su número de parámetros.

Bibliografía

[1] Paul W. Abrahams, *TeX for the Impatient*, Addison–Wesley, Reading, MA, 1990.
[2] *AMS Fonts User's Guide*, versión 2.2, American Mathematical Society, Providence, RI, 1995.
[3] Malcolm Clark, *A Plain TeX Primer*, Oxford University Press, Oxford, 1992.
[4] Victor Eijkhout, *TeX by Topic*, Addison–Wesley, Reading, MA, 1992.
[5] Michel Goossens, Frank Mittelbach & Alexander Samarin, *The* LaTeX *Companion*, Addison–Wesley, Reading, MA, 1994.
[6] Ronald L. Graham, Donald E. Knuth & Oren Patashnik, *Concrete Mathematics*, Addison–Wesley, Reading, MA, 1989.
[7] Eitan M. Gurari, *TeX and* LaTeX*: Drawing and Literate Programming*, McGraw-Hill, New York, 1994.
[8] Donald E. Knuth, *The TeXbook*, Addison–Wesley, Reading, MA, 1986.
[9] Donald E. Knuth, *TeX: The Program*, Addison–Wesley, Reading, MA, 1986.
[10] Leslie Lamport, LaTeX*: A Document Preparation System*, segunda edición, Addison–Wesley, Reading, MA, 1994.
[11] Raymond Seroul, *Le petit Livre de TeX*, InterEditions, Paris, 1989.
[12] Michael Spivak, *The Joy of TeX*, segunda edición, Addison–Wesley, Reading, MA, 1990.
[13] TUGboat: *Communications of the TeX User's Group*, TeX User's Group, Portland, Oregon; publicado cuatro veces al año.

Epílogo

Este documento fue concebido originalmente para un curso de entrenamiento en el uso de TEX, tanto a nivel básico como a nivel intermedio, que ofrecí en la Universidad de Costa Rica entre agosto y noviembre de 1990. En aquél tiempo, se componían los textos científicos con una gran variedad de procesadores de palabras, todos mas o menos decentes para la tarea cotidiana de escribir cartas, informes y artículos de periódicos. Los que apostamos a la emergencia de TEX como software universal para textos científicos debido, ante todo, a su portabilidad entre diversos sistemas de operación, eramos aún una minoría.

Decidí escribir, no un manual de fácil entrada para uso del usuario casual (hoy día abundan excelentes manuales de ese tipo), sino un breviario de referencia para quienes necesitaban el siguiente paso: acceso al lenguaje de programación de TEX. Por supuesto, el *TEXbook* ya cumple esa función, de una forma bastante definitiva, pero se rumoreaba que era de difícil lectura (fue un rumor que nunca logré confirmar). El curso presupone que su usuario quiere ahondar en el formato Plain TEX, descrito en el *TEXbook*.

Varios años después, TEX ahora es indispensable para cualquier comunicación a distancia en la ciencias físicas o en las matemáticas; con el crecimiento de la red Internet, ha logrado dominar el mercado de preprints científicos en forma absoluta. Basta echar una mirada al sitio `xxx.lanl.gov` o alguno de sus "espejos" nacionales. Con el crecimiento del CTAN (Comprehensive TEX Archive Network), se dispone de una enorme cantidad de información, de fácil acceso, sobre el buen uso de TEX, así que este curso ya no hace tanta falta. Para esta versión, modifiqué solamente aquéllos párrafos que presuponían el uso de una Macintosh con el programa *Textures* (que sigue siendo la mejor implementación integrada de TEX).

Algo ya previsible en 1990 era la emergencia del formato LATEX como sistema viable y flexible para quienes no deseen dedicar su escaso tiempo a detalles de presentación visual de sus ideas. Con el advenimiento de LATEX 2_ε, que ha logrado estandarizar y extender la variedad de estilos de documento disponibles, el Plain TEX ha pasado a segundo plano. Espero, sin embargo, que estas notas sean de alguna utilidad para quienes desean "levantar la tapa" para ver como funciona la maquinaria por dentro.

Joseph C. Várilly
`varilly@cariari.ucr.ac.cr`
San José, Diciembre de 1998

Indice Alfabético

→|, 2, 26, 37.
\→| ( ), 4.
↵, 2, 26, 32, 33, 58, 59.
\↵ ( ), 4.
␣, 26.
\␣ ( ), 4.
!‘ (¡), 3.
\" (¨), 3.
#, 26, 40, 43.
\# (#), 1.
$, 9, 13, 26, 31, 32.
$$, 9, 31, 32.
\$ ($), 1, 5.
%, 26, 36.
\% (%), 1.
&, 18, 26, 37, 40, 43.
&&, 42, 43.
\& (&), 1.
’ (’), 2.
’’ (”), 2.
\’ (´), 3.
\’{\i} (í), 20.
(, 12.
\((LaTeX), 4.
), 12.
\) (LaTeX), 4.
*, 36.
\+, 37, 38.
\,, 49.
- (-), 2.
-- (–), 2.
--- (—), 2.
\-, 34.
\. (˙), 3.
/, 12, 56.
\/, 8.
\1, 20.
:= (:=), 11.
\: (AmSTeX), 4.
\= (¯), 3, 4.
\>, 4, 20, 57.
?‘ (¿), 3.
@, 63.
[, 12.
\[(LaTeX), 4.
\, 1, 4, 26, 62.
\\ (AmSTeX), 4, 66.
], 12.
\] (LaTeX), 4.
^, 13, 26.
^^M, 58.
\^ (^), 3.
_, 13, 26.
_ (_), 39.
‘ (‘), 2.
‘‘ (“), 2.
\‘ (`), 3.
{, 26, 57, 59.
\{ ({), 12.
| (|), 10, 12.
\| (‖), 10, 12.
}, 26, 57.
\} (}), 12.
~, 20, 26, 30.
\~ (~), 3.

\AA (Å), 3.
\aa (å), 3.
Abrahams, Paul W., 70.
Abreviatura, 20, 30, 57.
\abstract (AmSTeX), 66.
\accent, 64.
Acento, 1, 3, 14, 57, 58, 64.
\active, 58.
\acute (´), 14, 64.
\address (AmSTeX), 66.
Admiración, 8.
\advance, 51, 52, 58, 60.
\advancepageno, 46–48, 58.
\AE (Æ), 3.
\ae (æ), 3.
\aleph (ℵ), 10.
Alfabeto griego, 10.
\aligned (AmSTeX), 66.
Alineamiento de ecuaciones, 18, 65.
Alineamiento bilateral, 19, 66.
\alpha (α), 10, 24, 44, 64, 65.
Altura, 28, 30, 42, 51, 53.
\amalg (⨿), 11, 13.
Ambiente, 58, 69.
American Mathematical Society, 1, 62, 62.
AMS Fonts User's Guide, 67, 70.
$\mathcal{AMS}$-LaTeX, 68.
amsppt.sty, 67.
amssym.tex, 67.
amssymb.sty, 68.
$\mathcal{AMS}$-TeX, 4, 32, 57, 62, 66.
amsthm.sty, 68.
Anchura, 28, 41, 51, 66.
\angle (∠), 10.
Anulación de tabuladores, 39.
Anulación de un macro, 57.
Apóstrofe, 2, 14.
\approx (≈), 10, 11.
\arccos (arccos), 17.
\arcsin (arcsin), 17.
\arctan (arctan), 17, 44.
Archivo auxiliar, 60, 65, 67, 69, 70.
Archivo de estilo, 67, 68.
Archivo de formato, 67.
Archivo externo, 60.
Archivo fuente, 7, 25, 52, 59, 61, 67–70.
\arg (arg), 17.
Argumento de un macro, 5, 22, 58, 59, 63.
Argumento opcional, 70.
Arroba, 26.
article.cls, 68.
Artículos matemáticos, 66.
ASCII, 25, 51, 58.
Asignación, 11, 48, 49, 54, 57, 58, 70.
Asignación global, 58.
\ast (∗), 11.
Asterisco, 26.
\asymp (≍), 11.
at, 6.
\atop, 16.
\author (AmSTeX), 66.
.aux, 69.

\b (_), 3.
\backslash (\ ), 10, 12.
\bar (¯), 14.
Barra corta, 2.
Barra larga, 2, 28.
Barra vertical, 49.
Base de datos, 60.
\baselineskip, 30, 46.
\Bbb (AmSTeX), 67.
\begin (LaTeX), 69, 70.

\beginsection, 63, 64.
\beta (β), 10.
\bf, 6–8, 24, 36, 39, 60, 64, 67.
\bgroup, 57, 65.
Big point, 26.
\bigbreak, 32, 35, 60.
\bigcap ($\bigcap$), 13.
\bigcirc ($\bigcirc$), 11.
\bigcup ($\bigcup$), 13.
\Biggl, 17.
\biggl, 17.
\Biggr, 17.
\biggr, 17.
\Bigl, 17.
\bigl, 17.
\bigodot ($\bigodot$), 13.
\bigoplus ($\bigoplus$), 13.
\bigotimes ($\bigotimes$), 13.
\Bigr, 17.
\bigr, 17.
\bigskip, 7, 29–32, 36, 45, 64.
\bigsqcup ($\bigsqcup$), 13.
\bigtriangledown ($\bigtriangledown$), 11.
\bigtriangleup ($\bigtriangleup$), 11.
\biguplus ($\biguplus$), 13.
\bigvee ($\bigvee$), 13.
\bigwedge ($\bigwedge$), 13.
Blackboard boldface, 67.
blue, 62.
Bodoni, Giambattista, 8.
\bold (AmSTeX), 67.
book.cls, 68.
\bot ($\bot$), 10.
\botmark, 61.
\bowtie ($\bowtie$), 11.
\Box (LaTeX), 69.
\box, 48–51, 54, 56.
bp, 6, 26, 50.
\break, 34, 35, 64.
\breve ($\breve{}$), 14.
\buildrel, 65.
\bullet ($\bullet$), 11.
by, 6, 51.
\bye, 7, 30, 62–64, 66.

\c (¸), 3.
Cabeza de página, 46.
Caja, 22, 28, 41, 48–51, 53.
Caja alta, 52.
Caja baja, 52.
Caja en blanco, 56.
Caja horizontal, 49, 53, 54.
Caja invisible, 42.
Caja vacía, 32, 46, 55, 64.
Caja vertical, 49, 53, 54, 65.
Cajas empiladas, 49.
\cal, 64.
Cambio matemático, 26.
Canal de lectura, 60.
Canal de salida, 60.
Caos, 63.
\cap ($\cap$), 11, 33.
Carácter, 51, 54, 57, 58, 63, 67.
Carácter activo, 20, 26, 58.
Carácter compuesto, 3.
Carácter de escape, 4, 26.
Carácter especial, 1.
Carácter ignorado, 26.
Carácter inválido, 26.
Carácter invisible, 2.
Carácter sobreescrito, 56.
Carácter visible, 1, 31.
\catcode, 58, 62, 65.
Categoría, 26, 58, 62.
cc, 6, 26, 50.
\CD (AmSTeX), 66.
\cdot ($\cdot$), 11.
\centerline, 6, 22, 30, 33, 36, 62, 64.
Centímetro, 26.
Centrar fórmulas, 53.
Centroide, 33.
\check ($\check{}$), 14.
Chequear sintaxis, 66.
\chi (χ), 10.
\choose, 16.
Cicero, 26.
\circ ($\circ$), 11.
Circuncentro, 33.
Cita textual, 32, 69.
\cleartabs, 39, 64.
\closein, 60.
\closeout, 60.
.cls, 68.
\clubsuit ($\clubsuit$), 10.
cm, 6, 26, 50.
cmbsy, 67.
cmbx, 64.
cmex, 9, 64.
cmmi, 9, 64.
cmmib, 67.
cmr, 8, 64.
cmsl, 64.
cmsy, 9, 62, 64.
cmti, 64.
cmtt, 64.
Códigos internos, 62.
Coeficiente Clebsch–Gordan, 23.
\colon (:), 12.
Columna, 37, 40, 42.
Columna derecha, 48, 49.
Columna izquierda, 48, 49.
Columna justificada, 40.
\columns, 39.
Coma, 8, 30.
Coma decimal, 27.
Comando, 1.
Comando primitivo, 62.
Comentario, 1, 7, 26.
Comillas, 2.
Computer Modern, 8, 27, 64–67.
Computer Modern Roman, 8.
Comunicación con la terminal, 59.
Concrete Mathematics, 67, 70.
\cong ($\cong$), 11.
Contador, 58, 69.
Contenido de un registro, 52.
Contenido de una caja, 56.
Contenido de una marca, 61.
Contenido de una página, 50.
Contractibilidad de goma, 29.
Contrarreferencia, 69.
"Coplas por la muerte de su padre", 33.
\coprod ($\coprod$), 13.
\copy, 54.
Corchete, 6, 12, 45, 63, 70.
Corrección itálica, 8.
Corte de página, 54, 57.
Corte de renglón, 34.
Corte obligado, 34.
\cos (cos), 17, 44, 65.
\cosh (cosh), 17.
\cot (cot), 17.
\coth (coth), 17.
\count, 49–51, 58, 70.
\cr, 18, 37, 38, 40, 66.
Crenaje, 37, 53, 54.
Crenaje vertical, 31.
\csc (csc), 17.
CTAN, 71.
Cuadratín, 5, 28.
Cuerpo de página, 28, 46.
\cup ($\cup$), 11.

\d (˙), 3, 57, 58.
\dag (†), 36.
\dagger (†), 11.

\dashv (⊣), 11.
\day, 25.
dd, 6, 26, 50.
\ddag (‡), 36.
\ddagger (‡), 11.
\ddot (¨), 14.
\def, 7, 20, 48, 57–59, 63, 66, 70.
\define (AmSTeX), 66, 70.
Definición de macros, 20–25, 57–60, 66, 70.
Definición local de macros, 40.
\deg (deg), 17.
\delcode, 62.
Delimitador, 12, 17, 63, 64.
\delimiter, 64.
\Delta (Δ), 10.
\delta (δ), 10, 57, 58.
Denominador, 15.
depth, 6.
Desempacar un registro, 51.
Designar un registro, 50, 63.
Despliegue, 14, 38, 45, 63, 69.
Despliegue multilineal, 18–20, 65, 66, 69.
\det (det), 17, 65.
Diagrama conmutativa, 66.
\diamond (⋄), 11, 33.
\diamondsuit (♢), 10.
Didot, Firmin, 8.
Didot, 26.
Diesis, 40.
Dígitos, 1, 26.
Dígitos arábigos, 51.
\dim (dim), 17.
\dimen, 49–51.
Dimensión, 26, 48.
\displaylines, 65.
\displaystyle, 16, 30, 44, 51, 64, 65.
\div (÷), 11.
\divide, 51.
División silábica, 65.
\document (AmSTeX), 66.
\documentclass (LaTeX), 52, 62, 68, 70.
\documentstyle (AmSTeX), 67.
\documentstyle (LaTeX), 69.
Dos puntos, 8, 12, 30.
\dosupereject, 46.
\dot (˙), 14.
\doteq (≐), 11.
\dotfill, 55, 64.
\dots (…), 30, 64.
\Downarrow (⇓), 12.
\downarrow (↓), 12.
\dp, 51, 56.
DraTEX, 62.
dummy, 67.
\dump, 62, 67.
dvips, 62.

Ecuaciones numeradas, 15, 19.
Efectos especiales, 65.
\egroup, 57, 65.
Eijkhout, Victor, 70.
\eject, 45, 55, 64.
Elipsis, 30.
\ell (ℓ), 10.
\else, 24, 46–48, 58, 60.
em, 6, 28, 50.
\emptyset (∅), 10.
Encabezado de una página, 45, 46, 61, 66, 68.
Encapsulated PostScript, 61.
\end, 62.
\end (LaTeX), 69, 70.
\endabstract (AmSTeX), 66.
\endaddress (AmSTeX), 66.
\endaligned (AmSTeX), 66.
\endauthor (AmSTeX), 66.
\endCD (AmSTeX), 66.
\enddocument (AmSTeX), 66.
\endhead (AmSTeX), 66.
\endinsert, 36, 65.
\endpmatrix (AmSTeX), 66.
\endref (AmSTeX), 66.
\endRefs (AmSTeX), 66.
\endsubhead (AmSTeX), 66.
\endtitle (AmSTeX), 66.
\endtopmatter (AmSTeX), 66.
\enskip, 29, 60.
\enspace, 24.
Enunciado de un teorema, 64.
Enviar una página a disco, 45–48.
EPlain, 62.
.eps, 61.
epsf.sty, 62.
epsf.tex, 62.
\epsilon (ϵ), 10, 58.
\eqalign, 18, 20, 53, 65, 66.
\eqalignno, 19, 65, 69.
\eqno, 15, 20.
\equiv (≡), 11.
Espaciamiento en fórmula, 9.
Espacio, 2, 25, 29.
Espacio en blanco, 4.
Espacio entre párrafos, 29.
Espacio entre renglones, 63.
Espacio explícito, 30.
Estilo de despliegue, 16.
Estilo de índices, 16.
Estilo de letra, 8.
Estilo de texto, 16.
Estirabilidad de goma, 29.
Eszet (ß), 3.
Etiqueta, 15, 19, 66.
\eta (η), 10.
eufm, 67.
Euler Fraktur Medium, 67.
\everypar, 52.
ex, 6, 28, 50.
\exists (∃), 10.
\exp (exp), 17.
\expandafter, 59.
Expansión de macros, 21, 57, 59, 60.
Exponente, 1, 13, 26, 65.
Exponentes dobles, 13.
Expresión matemática, 1.

Factor de magnificación, 27.
\fam, 64.
Familia de fuentes, 64.
Familia de símbolos matemáticos, 63.
Familia tipográfica, 8.
Fealdad, 35, 63.
\fi, 24, 46, 47, 58, 60.
Ficha, 22, 57.
fil, 6, 50, 55.
Fila, 40, 41.
fill, 6, 50.
filll, 6, 50.
Fin de grupo, 26.
Fin de línea, 26.
\firstmark, 61.
\fivebf, 64.
\fivei, 64.
\fiverm, 64.
\fivesy, 64.
\flat (♭), 10.
Flechas, 12, 13, 55.
Flecha larga, 65.
\folio, 47, 61, 65.
\font, 59, 64.
\footins, 65.
\footline, 46, 47.

\footnote, 36, 53, 58, 65.
\forall (∀), 10.
Formato, 62.
Formato de doble columna, 48.
Fórmulas matemáticas, 9, 64.
\frac, 67.
Fracción, 9, 15, 67.
\frak (AmSTeX), 67.
Fraktur, 3, 67.
Frases matemáticas, 35.
\frenchspacing, 30.
\frown (⌢), 11.
Fuente, 8, 64, 66, 67, 69.
Fuente abierta, 67.
Fuentes de la AMS, 69.
Fusión de columnas, 43.

\Gamma (Γ), 10.
\gamma (γ), 10.
\gcd (gcd), 17.
\gdef, 58, 59.
\ge (≥), 11.
\geq (≥), 11.
\gets (←), 12, 14, 55.
\gg (≫), 11.
.gif, 61.
\global, 58.
Goma, 28, 41, 49, 53.
Goma desechada, 55.
Goma entre columnas de una tabla, 45.
Goma vertical, 45.
\goodbreak, 35.
Goossens, Michel, 70.
\goth (AmSTeX), 67.
Graham, Ronald L., 67, 70.
Graphic Interchange Format, 61.
\grave (`), 14.
green, 62.
Grupo, 1, 6, 37, 57.
Guión, 2, 63.
Guión escondido, 35.
Guión opcional, 34.
Gurari, Eitan, 70.

\H (″), 3.
\halign, 40, 42, 44, 59, 65.
\hangafter, 33.
\hangindent, 33.
\hat (ˆ), 14.
\hbar (ℏ), 10.
\hbox, 22, 28–31, 35, 48, 50, 51, 53–56, 62.
\head (AmSTeX), 66.
\headline, 46, 47, 61.
\heartsuit (♡), 10.
height, 6.
Helvética, 8.
\hfil, 30, 34, 40–44, 47, 48, 55, 61.
\hfill, 7, 30, 38, 44, 55.
\hoffset, 46.
Hoja A4, 26.
\hom (hom), 17.
Hondura, 28, 30, 41, 51, 53.
\hookleftarrow (↩), 12.
\hookrightarrow (↪), 12, 57.
\hrule, 7, 28, 31, 32, 41–44, 53.
\hrulefill, 55, 64.
\hsize, 5, 22, 27, 30, 32, 45, 48, 51–54, 57, 62, 63.
\hskip, 29, 32, 49, 55.
\hss, 22, 30, 47, 48, 56, 62.
\ht, 51, 56.
hyphen.tex, 65.
\hyphenation, 34.

\i (ı), 20.
í, 20.
\ifcase, 25.
\ifdim, 24.
\ifeof, 60.
\iff (⟺), 12.
\ifhmode, 24.
\ifmmode, 24.
\ifnum, 24, 46, 58.
\ifodd, 24, 47.
\ifx, 24.
\Im (ℑ), 10.
\imath (ı), 10, 14.
\immediate, 60.
in, 6, 26, 50.
\in (∈), 10, 11.
Incentro, 33.
\indent, 32, 38.
índice de contenidos, 68, 69.
índices, 60.
\inf (inf), 17.
\infty (∞), 10, 51.
Inicio de grupo, 26.
IniTEX, 62.
\input, 7, 60, 65, 67.
Inserción flotante, 36, 45, 50, 53, 63, 65.
Inserción flotante, 37, 46, 51, 54, 65, 67.
\int (∫), 13.
Integral, 13.
Interlineación, 30, 46, 65.
Interrogación, 8.
\iota (ι), 10.
\it, 6, 8, 37, 39, 64, 67.
Itálico, 8.
\item, 33, 58, 64.
\item (LaTeX), 70.
\itemitem, 33, 64.

\jmath (ȷ), 10, 14.
\jobname, 60.
\joinrel, 65.
\jot, 63.
The Joy of TEX, 66, 70.
Justificación de párrafos, 54.
Justificación de renglones, 30, 33, 63.

\kappa (κ), 10.
\ker (ker), 17.
\kern, 37, 53.
Knuth, Donald Ervin, 1, 70.

\L (Ł), 3.
\l (ł), 3.
l, 62.
\Lambda (Λ), 10.
\lambda (λ), 10.
Lamport, Leslie, 70.
\land (∧), 11.
\langle (⟨), 12, 64.
LATEX, 32, 58, 60, 62, 67, 71.
LATEX 2.09, 68.
LATEX 2ε, 68, 71.
LATEX 3, 68.
latexsym.sty, 69.
\lbrace ({), 12.
\lbrack ([ ), 12.
\lceil (⌈), 12.
\le (≤), 11.
Leading, 30.
\leavevmode, 32.
\left, 17, 51.
\Leftarrow (⇐), 12.
\leftarrow (←), 12.
\leftarrowfill, 55.
\leftharpoondown (↽), 12.
\leftharpoonup (↼), 12.
\leftline, 33, 48, 64.
\Leftrightarrow (⇔), 12.
\leftrightarrow (↔), 12.

\leftskip, 32, 52, 69.
\leq (≤), 10, 11.
\leqalign, 20.
\leqalignno, 19, 65.
\leqno, 15.
\let, 57, 59.
Letra caligráfica, 9.
Letra gótica, 67.
Letra mayúscula, 4, 30, 52.
Letra minúscula, 4, 52.
Letra romana, 47, 65.
letter.cls, 68.
\lfloor (⌊), 12.
\lg (lg), 17.
\lgroup, 12, 17.
Liga, 1, 20, 30, 34, 35.
Ligadura, 2.
\lim (lim), 17.
\liminf (lim inf), 17.
\limsup (lim sup), 17.
\line, 7, 22, 30, 46, 53, 55, 64.
Línea de base, 28–30, 53.
Línea de puntos, 55.
Línea en blanco, 31, 31, 46.
\lineskip, 30, 42, 53.
\lineskiplimit, 30.
Lista de ítemes, 33, 58, 69.
Lista de fichas, 46, 49, 52, 59–61.
Lista de referencias, 66.
Lista horizontal, 28, 31, 53, 54.
Lista matemática, 54.
Lista vertical, 28, 31, 53.
Lista vertical principal, 45, 54, 55.
\ll (≪), 11.
\llap, 56.
Llaves, 6, 13, 48.
Llaves en un texto sustituto, 21.
Llaves redundantes, 23.
Llaves visibles, 12.
\lmoustache, 12, 17.
\ln (ln), 17, 44.
\lnot (¬), 10.
\loadbold (AmSTeX), 67.
\loadeufm (AmSTeX), 67.
\loadmsam (AmSTeX), 67.
\loadmsbm (AmSTeX), 67.
\log (log), 17.
Longitud, 6, 26, 48, 49.
\Longleftarrow (⟸), 12.
\longleftarrow (⟵), 12.
\Longleftrightarrow (⟺), 12.
\longleftrightarrow (⟷), 12.
\longmapsto (⟼), 12.
\Longrightarrow (⟹), 12, 65.
\longrightarrow (⟶), 12, 65.
\lor (∨), 11.
\lower, 53, 56.
\lowercase, 52.

\m@th, 63.
Macintosh, 1, 4, 71.
Macro, 20–25, 57, 59, 62.
Macro condicional, 24, 50.
Macro delimitado, 23.
Macro externo, 63.
Macro inaccesible, 63.
Macro local, 58.
Macro privado, 65.
Magnificación de tipos, 27.
\magnification, 27, 65.
\magstep, 27.
\magstephalf, 27.
\makefootline, 46.
\makeheadline, 46.
Manrique, Jorge, 33.
\mapsto (↦), 12.
Marca, 53, 61, 68.
Margen, 32, 37, 45, 69.
\mark, 53, 61.
\mathaccent, 64.
\mathchardef, 64, 67.
\mathcode, 62.
\mathop, 65.
\mathrel, 65.
\mathstrut, 42.
Math unit, 49.
Matitálico, 9, 64.
\matrix, 18, 65.
Matriz, 18, 65, 66.
\max (max), 17.
Mayúsculas, 59.
McGraw-Hill, 56.
\medbreak, 24, 35.
\medskip, 7, 29, 32, 45, 60, 64.
Mensajes de ayuda, 63.
\message, 59, 60.
METAFONT, 8, 66.
Microsoft Word, 26.
\mid (|), 11.
\midinsert, 36, 53, 65.
Milímetro, 26.
\min (min), 17.
minus, 6, 29, 50.
\mit, 64.
Mittelbach, Frank, 70.
\mkern, 65.
mm, 6, 26, 50.
\models (⊨), 11.
Modo de compatibilidad (LaTeX), 69.
Modo horizontal, 31, 54, 56, 64, 65.
Modo horizontal restringido, 31.
Modo matemático, 9, 13, 31, 49, 53–56, 63–65, 67.
Modo matemático de despliegue, 9, 31.
Modo vertical, 31, 36, 41, 55, 56.
Modo vertical interno, 31.
\month, 25.
\moveleft, 56.
\moveright, 56.
\mp (∓), 11.
msam, 67.
msbm, 67.
\mskip, 49.
mu, 6, 49, 50.
\mu (μ), 10.
Mugoma, 49.
\multiply, 51.
\multispan, 43.
\muskip, 49.

\nabla (∇), 10.
\narrower, 32, 64.
\natural (♮), 10.
\ne (≠), 11.
\nearrow (↗), 12.
\neg (¬), 11.
Negrilla, 6, 8.
\neq (≠), 11, 20.
\newbox, 48–51, 54, 63.
\newcommand (LaTeX), 70.
\newcount, 50, 58, 63.
\newdimen, 48, 50, 51, 63.
\newenvironment (LaTeX), 69.
\newfam, 63.
\newhelp, 63.
\newif, 48, 50, 63.
\newinsert, 63, 65.
\newlanguage, 63.

\newmuskip, 50, 63.
\newread, 60, 63.
\newskip, 50, 51, 63.
\newtoks, 50, 63.
\newwrite, 60, 63.
\ni (∋), 11.
\noalign, 41, 44.
\nobreak, 35, 60, 64.
\noexpand, 59, 60.
\noindent, 24, 32, 60.
\nointerlineskip, 30, 46.
\nolimits, 65.
Nombres y apellidos, 35.
\nonfrenchspacing, 30, 65.
\nopagenumbers, 46, 47, 65.
\normalbaselines, 65.
\not (⁄), 11.
Nota al pie de página, 36, 45, 50, 58, 65.
\notin (∉), 11.
\nu (ν), 10.
\null, 55.
\number, 25, 47.
Numeración automática, 58.
Numeración hexadecimal, 63.
Numerador, 15.
Número arábigo, 47.
Número de una ecuación, 15, 19, 69.
Número de página, 45, 47, 50.
Número entero, 49–51.
Número romano, 47.
\nwarrow (↖), 12.

ñ, 5.

\O (Ø), 3.
\o (ø), 3.
\obeylines, 33, 58, 64.
Oblicuo, 8.
\odot (⊙), 11.
\OE (Œ), 3.
\oe (œ), 3.
\of, 13.
\offinterlineskip, 42, 44.
\oint (∮), 13.
\Omega (Ω), 11.
\omega (ω), 10.
\ominus (⊖), 11.
Omisión de plantilla, 41, 43.
\omit, 41, 43, 44.
\openin, 60.
\openout, 60.
Operación binaria, 11, 63.
Operación aritmética con registros, 51.
Operador grande, 13, 65.
\oplus (⊕), 11.
\or, 25.
Ortocentro, 33.
\oslash (⊘), 11.
\otimes (⊗), 10, 11.
\outer, 63.
\output, 46, 48, 65.
\outputpenalty, 46.
\over, 15, 17, 44, 65.
\overfullrule, 52, 63.
\overleftarrow ($\overleftarrow{\ }$), 13, 14.
\overline ($\overline{\ }$), 13, 14.
\overrightarrow ($\overrightarrow{\ }$), 13, 14.
\owns (∋), 11.

\pagebody, 46, 48.
\pageinsert, 36, 65.
\pageno, 5, 24, 47, 50–52, 58, 60.
Palabra clave, 6, 26.
Palabra de control, 4, 5, 9, 57, 63.
\par, 24, 31, 32, 35, 54, 55, 58, 60, 62.
\parallel (∥), 11.
Parámetro, 1, 22, 26, 46, 58, 63, 67, 70.
Parámetro de formato, 5.
Paréntesis, 6, 12, 42, 49, 63.
\parindent, 28, 32, 52, 54, 63.
Párrafo, 28, 31, 52, 64.
Párrafo interrumpido, 32.
\parskip, 51.
Partes nombradas de un documento, 35.
\partial (∂), 10, 57.
Pascal, 49.
Patashnik, Oren, 70.
pc, 6, 26, 50.
Penalty, 35, 41, 53, 54.
\penalty, 35, 53, 55, 62.
Penalty desechado, 55.
\perp (⊥), 11.
\phantom, 56, 65.
\Phi (Φ), 10.
\phi (ϕ), 10.
\Pi (Π), 10, 13.
\pi (π), 10.
Pica, 26.
PiCTeX, 62.
Pie de página, 46, 47.
Pila de cajas, 53.
Plain TeX, 4, 7, 20, 28–32, 45, 50, 55–58, 62, 66–68, 70, 71.
plain.tex, 63–65.
\plainoutput, 46, 48, 65.
Plantilla, 40, 45.
plus, 6, 29, 50, 55.
\pm (±), 11.
\pmatrix, 18, 21, 56, 65.
\pmatrix (AmSTeX), 66.
Posiciones de tabuladores, 37.
PostScript, 8, 26, 27, 67.
\Pr (Pr), 17.
Preámbulo, 40, 41, 59, 70.
\prec (≺), 11.
\preceq (⪯), 11.
Preposición, 35.
Preprint, 67.
\pretolerance, 54, 63.
Prima, 12, 14.
\prime (′), 10, 12, 14.
\proclaim, 23, 63, 64.
\prod (∏), 13.
\propto (∝), 11.
Proyecto Euler, 67.
.ps, 61.
\Psi (Ψ), 4, 10.
\psi (ψ), 4, 10.
pt, 6, 26, 49–52, 55.
Pulgada, 26, 32.
Puntal, 41, 46.
Punto, 8, 26.
Punto de Gergonne, 33.
Punto de referencia, 28, 53, 56.
Punto decimal, 27, 41.
Punto y coma, 8, 30.
Puntos suspensivos, 30.
Puntuación, 8, 10, 12, 26, 30, 65.

\qquad ( ), 38, 64.
\quad ( ), 5, 28, 32, 38, 40, 43, 44, 66.
Quéeseso, 53, 61.

\raggedright, 33, 64.
\raise, 56.
Raíz cuadrada, 13.
Raíz cúbica, 13.
\rangle (⟩), 12.
\rbrace (}), 12.
\rbrack ( ]), 12.
\rceil (⌉), 12.

\Re ($\Re$), 10.
\read, 59, 60.
red, 62.
\redefine (AmSTeX), 66.
\ref (AmSTeX), 66.
Referencias, 60, 66, 69.
\Refs (AmSTeX), 66.
Registro, 49, 58, 65.
Registro de caja, 54.
Regla, 28, 53.
Regla horizontal, 28, 41, 55.
Regla vertical, 28, 42.
Relación, 11.
Relación compuesta, 11, 65.
\relax, 29.
\Relbar ($=$), 65.
\relbar ($-$), 65.
\renewcommand (LaTeX), 70.
Renglón, 28, 29, 45.
Renglón modelo, 38.
report.cls, 68.
Resorte, 30, 41, 43, 44, 49, 53, 55.
Resorte horizontal, 30.
Resorte vertical, 30.
Resorte visible, 55, 64.
Retorno de carro, 2, 25, 58.
Retroceso, 65.
Retroceso no destructivo, 56.
\rfloor ($\rfloor$), 12.
\rgroup, 12, 17.
\rho (ρ), 10.
\right, 17, 51.
\Rightarrow ($\Rightarrow$), 12, 65.
\rightarrow ($\rightarrow$), 12, 57, 65.
\rightarrowfill, 55.
\rightharpoondown ($\rightharpoondown$), 12.
\rightharpoonup ($\rightharpoonup$), 12.
\rightleftharpoons ($\rightleftharpoons$), 12.
\rightline, 33, 64.
\rightskip, 32, 69.
\rlap, 40, 56.
\rm, 8, 10, 64–67.
\rmoustache, 12, 17.
Rocicki, Tom, 62.
\roman (AmSTeX), 67.
\romannumeral, 47.
Romano, 8.
\root, 13.
Rutina de salida, 45–48, 61, 65–68.

Samarin, Alexander, 70.
Sangría, 28, 32, 37, 56.
Sangría colgante, 33, 70.
Sanserif, 8.
scaled, 6.
Scaled point, 26.
\scriptfont, 64.
\scriptscriptfont, 64.
\scriptscriptstyle, 16, 64.
\scriptstyle, 16, 64.
\searrow ($\searrow$), 12.
\sec (sec), 17, 44.
Secar la goma, 29.
Sección, 66, 68.
Seña de referencia, 36.
Seña flotante, 33, 70.
Serifes, 8.
Seroul, Raymond, 70.
\setbox, 48, 49, 51, 54, 56.
\setminus ($\setminus$), 11.
\settabs, 38, 39, 64.
\sevenbf, 64.
\seveni, 64.
\sevenrm, 47, 64.
\sevensy, 64.
\sharp ($\sharp$), 10.
\shipout, 46, 48, 60.
\showthe, 52.
\Sigma (Σ), 10, 13.
\sigma (σ), 10.
Signo aritmético, 1.
Signo de admiración inicial, 3.
Signo de dólar, 31.
Signo de interrogación inicial, 3.
Signo de puntuación, 1.
Signo menos, 2, 9, 63.
\sim ($\sim$), 11.
Símbolo aritmético, 26.
Símbolo de control, 4.
Símbolo matemático, 9, 35, 63.
Símbolo ordinario, 10, 64.
Símbolos en serie, 35.
\simeq ($\simeq$), 11.
\sin (sin), 17, 44.
\sinh (sinh), 17.
Sintaxis, 68, 70.
Sintaxis funcional, 23.
\skip, 49, 52.
\sl, 7, 8, 24, 37, 64, 67.
\smallbreak, 35.
\smallint ($\smallint$), 13.
\smallskip, 29–33, 35–37, 41, 45, 51, 58, 64.
\smallskipamount, 51, 63.
\smash, 56, 65.
\smile ($\smile$), 11.
sp, 6, 26, 50.
\spadesuit ($\spadesuit$), 10.
\special, 53, 61.
Spivak, Michael David, 70.
spread, 6, 50, 55.
\sqcap ($\sqcap$), 11.
\sqcup ($\sqcup$), 11.
\sqrt ($\sqrt{\ }$), 13, 42.
\sqsubseteq ($\sqsubseteq$), 11.
\sqsupseteq ($\sqsupseteq$), 11.
\ss (ß), 3.
Stanford University, 67.
\star ($\star$), 11.
\strut, 41, 44, 64.
.sty, 68.
\subhead (AmSTeX), 66.
Subíndice, 1, 13, 26, 65.
Subíndices dobles, 13.
Subir o bajar una caja horizontal, 53.
\subsection (LaTeX), 68.
\subset ($\subset$), 11.
\subseteq ($\subseteq$), 11.
\succ ($\succ$), 11.
\succeq ($\succeq$), 11.
\sum ($\sum$), 13, 51.
Suma, 13.
\sup (sup), 17.
Suprimir una expansión, 59.
\supset ($\supset$), 11.
\supseteq ($\supseteq$), 11.
\surd ($\surd$), 10.
Surdo, 13.
\swarrow ($\swarrow$), 12.
\syntax (AmSTeX), 67.

\t ($\widehat{\ }$), 3.
Tabla reglada, 41, 44.
\tabskip, 45.
Tabulación, 2, 25, 26, 37, 64.
Tabulador, 1, 26, 37.
Tamaño aparente de caja, 56.
Tamaño natural de goma, 29.
\tan (tan), 17, 44.
\tanh (tanh), 17.
\tau (τ), 10.
Tecla de escape, 4.
Teclado, 1, 25.
Tecleo de espacios, 9.
\tenbf, 64.
\tenex, 64.

\teni, 64.
\tenit, 64.
\tenrm, 47, 61, 64.
\tensl, 64.
\tensy, 64.
\tentt, 64.
Terminal, 52, 60.
\TeX (TEX), 4.
.tex, 60.
The TEXbook, 1, 22, 35, 39, 50, 54, 55, 58–62, 71.
TeX log, 7, 45, 52, 59, 67.
TEX: The Program, 49.
TEX User's Group, 70.
\textfont, 64.
\textstyle, 16, 64.
Textures, 1, 4, 27, 46, 62, 71.
Texto en colores, 62.
Texto equilibrado, 21, 59.
Texto paramétrico, 23.
Texto sustituto, 20, 57, 59, 63, 70.
\the, 46, 51, 58, 60.
\Theta (Θ), 10.
\theta (θ), 10.
\tilde (˜), 14.
Times, 8.
\times (×), 11.
Tipo de teclado, 8.
\title (AmSTeX), 66.
Título de artículo, 66.
Título de sección, 64, 66.
to, 6, 44, 50, 53, 55, 59.
\to (→), 12, 14, 17, 55, 57.
.toc, 69.
\today, 25, 33.
\toks, 49, 51.
\tolerance, 54, 63.
\top (⊤), 10.
\topins, 65.
\topinsert, 36, 53, 65.
\topmark, 61.
\topmatter (AmSTeX), 66.
\topskip, 47.
Traslapo de texto, 47, 56.
Trazo vertical, 42.
\triangle (△), 10.
\triangleleft (◁), 11.
\triangleright (▷), 11.
Triángulo, 33.
true, 6, 27.
\tt, 8, 64, 67.
\typeout (LaTeX), 60.

\u (˘), 3.
\undefined, 57.
\underline (_), 13, 56.
Unidad lectora, 3.
Unidades SI, 37.
\Uparrow (⇑), 12.
\uparrow (↑), 12.
\Updownarrow (⇕), 12.
\updownarrow (↕), 12.
\uplus (⊎), 11.
\uppercase, 52, 59.
\Upsilon (Υ), 10.
\upsilon (υ), 10.
\UseAMSsymbols (AmSTeX), 67.
\usepackage (LaTeX), 68.

\v (ˇ), 3.
Vara, 4.
\varepsilon (ε), 10.
Variables, 9.
\varphi (φ), 10.
\varpi (ϖ), 10.
\varrho (ϱ), 10.
\varsigma (ς), 10.
\vartheta (ϑ), 10.
\vbox, 28–31, 38–40, 42, 44–46, 48, 50, 53, 54, 57, 65.
\vcenter, 50, 53, 54, 65.
\vdash (⊢), 11.
\vec (⃗), 14.
\vee (∨), 11.
Versalita, 8.
\Vert (‖), 10, 12.
\vert (|), 10, 12.
\vfil, 30.
\vfill, 30, 32, 45, 55, 62.
VirTEX, 62.
\voffset, 46, 47.
\vrule, 28, 32, 42, 44.
\vsize, 5, 45, 63.
\vskip, 7, 27, 29, 31, 32, 36, 46, 49, 51.
\vss, 30, 46, 57.
\vtop, 50, 53, 54.

\wd, 51, 56.
\wedge (∧), 11.
Whatsit, 53.
\widehat (̂), 14.
\widetilde (̃), 14.
width, 6.
\wp (℘), 10.
\wr (≀), 11.
\write, 53, 60, 69.

\Xi (Ξ), 10.
\xi (ξ), 10.
xxx.lanl.gov, 71.
XY-pic, 62.
\year, 25.
\zeta (ζ), 10.

Indice de Contenidos

1 Los Caracteres de TEX 1
1.1 Caracteres del teclado 1
1.2 Caracteres de texto impreso 2
1.2.1 Comillas 2
1.2.2 Guiones 2
1.2.3 Ligaduras 3
1.3 Caracteres compuestos especiales 3
1.4 Acentos ortográficos 3

2 Comandos en TEX 4
2.1 Símbolos de control 4
2.2 Palabras de control 4
2.3 Efectos de los comandos 5
2.3.1 Cambios ambientales 5
2.3.2 Comandos que toman argumentos . . 5
2.3.3 Comandos autónomos 5
2.4 Parámetros de Plain TEX 5
2.5 Palabras claves 6
2.6 Agrupamiento 6
2.7 Un archivo fuente ejemplar 7

3 Tipos y Tamaños de letra 8
3.1 Tipos Computer Modern 8
3.2 Cambios de Estilo 8
3.2.1 La corrección itálica 8
3.3 Estilos matemáticos 9

4 Fórmulas matemáticas 9
4.1 El bestiario de símbolos 10
4.1.1 Caracteres matemáticos 10
4.1.2 Letras griegas 10
4.1.3 Símbolos ordinarios 10
4.1.4 Operaciones binarias 11
4.1.5 Relaciones 11
4.1.6 Flechas 12
4.1.7 Puntuación 12
4.1.8 Delimitadores 12
4.1.9 Operadores grandes 13
4.2 Raíces y barras 13
4.3 Subíndices y exponentes 13
4.4 Acentos en modo matemático 14
4.5 Fórmulas desplegadas 14
4.6 Fracciones 15
4.6.1 Tamaños de letras en fórmulas 16
4.7 Funciones matemáticas 16
4.8 Delimitadores 17
4.8.1 Delimitadores `\big` y `\bigg` 17
4.8.2 Los comandos `\left` y `\right` 17
4.9 Matrices 18
4.10 Alineamiento de ecuaciones 18
4.10.1 Alineamientos no numerados 18
4.10.2 Alineamientos con etiquetas 19

5 Macros 20
5.1 Abreviaturas simples 20
5.2 Macros con un parámetro 22
5.3 Macros con dos o más parámetros 22
5.4 Macros delimitados 23
5.5 Macros condicionales 24

6 El modo de funcionar de TEX 25
6.1 La unidad lectora de TEX 25
6.2 Dimensiones 26
6.2.1 Unidades de longitud 26
6.2.2 Magnificación de tipos 27
6.2.3 Dimensiones absolutas 27
6.2.4 Medidas que dependen del tipo 28
6.3 Cajas 28
6.4 Goma 28
6.4.1 Flexibilidad de la goma 29
6.4.2 Brincos verticales entre párrafos . . . 29
6.4.3 Resortes 29
6.4.4 Goma después de puntuación 30
6.4.5 Interlineación 30
6.5 Modos 31

7 Formatos de texto 31
7.1 Párrafos 31
7.2 Sangría 32
7.3 Los márgenes 32
7.4 Sangría colgante 33
7.5 Justificación de renglones 33
7.6 Cortes de renglón 34
7.6.1 Cortes obligados 34

7.6.2 Guiones opcionales 34
7.6.3 Ligas 35
7.6.4 Guiones escondidos en cajas 35
7.6.5 Penalties 35
7.7 Notas al pie de página 36
7.8 Inserciones de bloques flotantes 36

8 Tabulación y Tablas 37
8.1 Tabulación 37
8.2 Colocación de tabuladores 38
8.2.1 Tabulación centrada 38
8.2.2 Uso de renglones modelos 38
8.2.3 Columnas de igual anchura 39
8.2.4 Anulación de tabuladores 39
8.3 Alineamiento con plantillas 40
8.3.1 Plantillas con caracteres 40
8.3.2 Entradas excepcionales 41
8.4 Tablas regladas 41
8.4.1 Reglas horizontales en tablas 41
8.4.2 Reglas verticales en tablas 42
8.5 Diversos recursos para hacer tablas . . . 43
8.5.1 Omisión de plantillas 43
8.5.2 Fusión de columnas 43
8.5.3 Plantillas repetidas 43
8.5.4 Una tabla reglada completa 44
8.6 Resortes en las tablas 44

9 Rutinas de Salida 45
9.1 La rutina de salida de Plain TEX 45
9.2 Encabezados y pies 46
9.2.1 Colocación de encabezados y pies . 46
9.2.2 Modificación de encabezados y pies 47
9.2.3 Números arábigos y romanos 47
9.3 Formato de doble columna 48

10 Registros 49
10.1 Los registros de TEX 49
10.2 Designación de registros 50
10.3 Aritmética con registros 51
10.3.1 Aritmética con cajas 51
10.4 Impresión de contenidos de registros . 51
10.4.1 Conversión de caja de letra 52

11 Manejo de cajas 53
11.1 Cajas empiladas y enfiladas 53
11.1.1 Cajas verticales 53
11.1.2 Cajas horizontales 53
11.1.3 Listas verticales y horizontales . . . 53
11.2 Párrafos en cajas verticales 54
11.3 Cómo guardar y copiar cajas 54
11.4 Uso de resortes en cajas 55
11.4.1 Traslapos de texto 55
11.5 Manipulación de cajas 56

12 Macros avanzados 57
12.1 Definición con `\def` y `\let` 57
12.2 Macros locales y globales 58
12.3 Macros con contadores 58
12.4 Expansión de macros 59
12.5 Comunicación con archivos externos . 59
12.5.1 Comunicación con la terminal 59
12.5.2 Lectura de archivos externos 60
12.5.3 Escritura en archivos externos 60
12.5.4 Comunicación con rutinas de salida 61
12.6 Incorporación de archivos gráficos . . . 61

13 Archivos de Formato 62
13.1 Introducción a Plain TEX 62
13.1.1 Códigos 62
13.1.2 Designación de registros 63
13.1.3 Parámetros 63
13.1.4 Fuentes 64
13.1.5 Macros para texto 64
13.1.6 Macros para matemáticas 64
13.1.7 Macros para la rutina de salida . . 65
13.1.8 Otros macros 65
13.2 Introducción al $\mathcal{AMS}$-TEX 66
13.2.1 Organización del documento 66
13.2.2 Formateo de fórmulas 66
13.2.3 Fuentes de la AMS 66
13.2.4 Chequeo de sintaxis 67
13.2.5 Archivos de estilo 67
13.3 Introducción a LATEX 67
13.3.1 Archivos de estilo 68
13.3.2 Archivos auxiliares 69
13.3.3 Ambientes 69
13.3.4 Definición de macros en LATEX . . . 70
Bibliografía 70
Epílogo 71
Indice Alfabético 72