Desarrollo de herramientas LaTeX 2_ε para composición avanzada de textos científicos
Development of LaTeX 2_ε Tools for Advanced Typesetting of Scientific Texts.

Gonzalo Medina Arellano

Universidad Nacional de Colombia
Sede Manizales
Facultad de Ciencias Exactas y Naturales
Manizales, Colombia
2012

Desarrollo de herramientas LaTeX 2_ε para composición avanzada de textos científicos
Development of LaTeX 2_ε Tools for Advanced Typesetting of Scientific Texts.

Tesis elaborada por

Gonzalo Medina Arellano

Presentada a la Facultad de Ciencias Exactas y Naturales

de la Universidad Nacional de Colombia - Sede Manizales

como Requisito Parcial para

obtener el Grado de

Magíster en Ciencias – Matemática Aplicada

Director
Carlos Daniel Acosta Medina

Codirector
Fabián Fernando Serrano Suárez

Departamento de Matemáticas y Estadística

Universidad Nacional de Colombia - Sede Manizales

Enero de 2012

en Memoria de
Rosalba Niño de Castelblanco (1 931 – 2 011)

Agradecimientos

Agradezco en primer lugar a los profesores Carlos Daniel Acosta Medina y Fabián Fernando Serrano Suárez y a la Facultad de Ciencias Exactas y Naturales de la Universidad Nacional de Colombia, Sede Manizales, por su apoyo en la realización de esta Maestría. Agradezco también a toda la comunidad de desarrolladores y usuarios de LaTeX; en particular, quiero expresar mis agradecimientos a Martin Scharrer, Christophe Caignaert y Lucero Álvarez Miño por sus comentarios y sugerencias sobre mejoras posibles a los paquetes. Por último, y de manera muy personal agradezco a mi amada compañera Mónica Lucía Castelblanco Niño por su incondicional cariño y su infinita paciencia.

Resumen

En este trabajo se presenta el desarrollo de mejoras a dos paquetes LaTeX 2ε creados e implementados por el autor y que aportan soluciones novedosas y previamente inexistentes a problemas de composición avanzada de textos: el paquete `background` permite la ubicación de material arbitrario (texto, imágenes, etc.) en todas o en algunas páginas de un documento, dando al usuario control preciso sobre los atributos (ubicación, color, opacidad, etc.) del material a ser presentado; el paquete `fancypar` permite decorar párrafos individuales de texto utilizando cinco estilos predefinidos; el usuario puede, además, fácilmente definir sus propios estilos. Los dos paquetes ya hacen parte de las distribuciones estándar del sistema LaTeX. Las mejoras fueron realizadas para brindar mayor flexibilidad a los paquetes, incrementando su funcionalidad y mejorando la interacción con el usuario.

Palabras clave: TeX, LaTeX 2ε, programación avanzada, paquete, `background`, `fancypar`.

Abstract

In this work we present improvements made to two LaTeX 2ε packages created by the author; these packages provide novel and previously inexistent solutions to some problems of advanced text composition: the `background` package allows the placement of arbitrary material (text, images, etc.) in all or in some pages of a document, giving the user precise control over the attributes (contents, position, color, opacity) of the background material that will be displayed; the `fancypar` package decorates individual paragraphs of a document, offering five pre-defined styles; the user may easily define customized styles. Both packages are included in the main distributions of the LaTeX system. The improvements were made in order to provide more flexibility, to enhance the packages functionality and to improve the interaction with the user.

Keywords: TeX, LaTeX 2ε, advanced programming, package, `background`, `fancypar`.

Índice general

Resumen i

Índice de cuadros iv

Índice de figuras v

Introducción 1

1. Elementos de programación avanzada en TEX y en LATEX 3
- 1.1. Definiciones simples en TEX y en LATEX . . . 3
  - 1.1.1. Definición de comandos en TEX y en LATEX . . . 3
  - 1.1.2. Definición de entornos en LATEX . . . 9
  - 1.1.3. Definición y manipulación de contadores en LATEX . . . 11
- 1.2. Definición simultánea de un comando y un entorno . . . 13
- 1.3. Condicionales . . . 15
  - 1.3.1. Forma que toman los condicionales en TEX . . . 15
  - 1.3.2. Algunos de los condicionales . . . 15
- 1.4. Cómo realizar copias de comandos . . . 19

2. El paquete `background` 22
- 2.1. Descripción del paquete . . . 22
  - 2.1.1. Información general . . . 23
- 2.2. Implementación actual del paquete . . . 23
  - 2.2.1. Ejemplos de uso del paquete en su versión actual . . . 27
- 2.3. Cambios sugeridos . . . 28
- 2.4. La nueva versión del paquete . . . 29
  - 2.4.1. Estrategia de implementación de los cambios . . . 29
  - 2.4.2. Código comentado de la nueva versión del paquete . . . 30
  - 2.4.3. Ejemplos de uso de la nueva versión del paquete . . . 34

3. El paquete `fancypar` 38
- 3.1. Descripción del paquete . . . 38
  - 3.1.1. Información general . . . 39
- 3.2. Implementación actual del paquete . . . 39
  - 3.2.1. Ejemplos de uso del paquete en su versión actual . . . 46
- 3.3. Cambios sugeridos . . . 52
- 3.4. La nueva versión del paquete . . . 53
  - 3.4.1. Estrategia de implementación de los cambios . . . 53
  - 3.4.2. Código comentado de la nueva versión del paquete . . . 54
  - 3.4.3. Ejemplos de uso de la nueva versión del paquete . . . 62

A. El código de las nuevas versiones de los paquetes `background` y `fancypar` **67**

Bibliografía **68**

Índice de cuadros

1.1. Comandos TEX asociados a condicionales 16

2.1. Comandos de usuario del paquete background que permiten modificar los atributos del material de fondo 24
2.2. Las diferentes opciones del paquete background en su actual versión 25
2.3. Las opciones disponibles en la nueva versión del paquete background, sus posibles valores y los valores por defecto 35

3.1. Resumen de los estilos predefinidos en la actual versión del paquete fancypar, junto con sus opciones y los valores por defecto 47
3.2. Resumen de los estilos predefinidos en la nueva versión del paquete fancypar, junto con sus opciones y los valores por defecto 63

Índice de figuras

2.1. Esquema personalizado de numeración con el paquete `background` 37
2.2. Diferente material de fondo para las páginas pares e impares utilizando el paquete `background` 37

3.1. Ejemplo de aplicación simultánea a varios párrafos del nuevo entorno `ZebraPar`, para el estilo «cebra» del paquete `fancypar`. Los nuevos entornos admiten material de tipo `verbatim` 65
3.2. Ejemplo de aplicación simultánea a varios párrafos del nuevo entorno `MarkedPar` junto con una de sus nuevas opciones, para el estilo «párrafo marcado» del paquete `fancypar` 66

Introducción

TEX es un sistema de composición de textos escrito por Donald E. Knuth, quien en el prefacio de su TEXbook escribió «(TEX is) intended for the creation of beautiful books — and especially for books that contain a lot of mathematics». En la actualidad, el uso de TEX ha trascendido el dominio de los textos científicos con alto contenido matemático y se ha convertido en un poderoso sistema de composición de textos en las más diversas disciplinas (literatura, teología, lingüística); en todos los casos, sigue siendo válida la premisa con que su creador lo concibió: la creación de libros hermosos.

Knuth es Profesor Emérito de la Universidad de Stanford en California, USA. Knuth desarrolló la primera versión de TEX en 1 978 para escribir su obra «the Art of Computer Programming». La idea inicial contó con gran acogida y Knuth produjo una segunda versión (en 1 982) que es la base del sistema usado hoy en día.

Knuth además desarrolló un sistema de «programación literaria» para escribir el código de TEX, y el código literario (WEB) de TEX está disponible de manera gratuita, junto con las herramientas necesarias para procesar el código fuente y producir código compilable e incluso una versión impresa; en principio no existen misterios acerca de cómo funciona TEX. Adicionalmente, el sistema WEB permite portar TEX a sistemas operativos nuevos. Dos de las características más importantes de TEX (aparte de su funcionalidad, claro está) son su portabilidad y el ser de código abierto y gratuito.

Hacia 1 984 Leslie Lamport desarrolló LATEX, que es una colección de macros TEX que facilita su uso. LATEX brinda la posibilidad de evitar que el usuario deba preocuparse por aspectos de presentación del documento y se concentre tan solo en el contenido y en la estructura lógica del mismo. Utilizando clases y paquetes, un mismo documento puede procesarse para obtener una gran variedad de diseños.

Una de las fortalezas de TEX y de LATEX es que son lenguages de programación Turing-completos, lo cual permite al programador experto realizar algoritmos para automatizar la realización de tareas tipográficas complejas.

La última versión del sistema LATEX de Lamport (LATEX2.09, de 1 992) fue reemplazada en 1 994 por una nueva versión desarrollada por el Equipo LATEX y denominada LATEX 2_ε. LATEX 2_ε es la versión de LATEX que actualmente se utiliza. En estos momentos, el Equipo LATEX se encuentra desarrollando la próxima versión (LATEX3) del sistema LATEX (de hecho, los sistemas LATEX actuales incluyen ya muchos de los paquetes y formatos de LATEX3). Adicionalmente, dos nuevos formatos se encuentran en fuerte desarrollo: LuaTEX y ConTEXt.

Aunque cualquier aspecto de la composición de textos puede modificarse y controlarse utilizando LaTeX 2ε (o incluso TeX), es útil para el usuario final contar con paquetes que pongan a su disposición macros amistosos que le eviten tener que recurrir a programación de bajo nivel para obtener las modificaciones deseadas para sus documentos. En la actualidad, LaTeX 2ε está «congelado»; es decir, no se admiten modificaciones al kernel (los únicos cambios permitidos al kernel provienen de reportes de «bichos»), así que el avance de LaTeX 2ε lo impulsa la creación de nuevos paquetes.

Existen en el momento alrededor de 2 000 paquetes en CTAN que hacen posible controlar muchos aspectos de la creación de documentos, desde aquellos macrotipográficos como marginado, pies de página, de imagen o de cuadro, etc. hasta aquellos de microtipografía como ligaduras, interletrado, interlineado, etc.

El presente trabajo consiste en el desarrollo, diseño e implementación de versiones mejoradas de dos de tales paquetes: `background` y `fancypar`. El paquete `background` permite colocar cualquier tipo de material (ya sea texto, imágenes, gráficas creadas por el usuario, etc.) al estilo de marca de agua, en cualquier posición dentro de algunas o de todas las páginas de un documento. El usuario tiene control sobre los atributos del material que será colocado: contenido, tamaño, posición, color, opacidad, etc. El paquete `fancypar` permite decorar párrafos individuales de texto de un documento; el paquete ofrece también una serie de comandos de usuario para que éste pueda fácilmente definir sus propios estilos.

Los dos paquetes y las mejoras propuestas han sido concebidos teniendo en cuenta el espíritu que Donald E. Knuth y Leslie Lamport tuvieron en mente al desarrollar TeX y LaTeX: la producción de textos de calidad tipográfica profesional.

La manera en que está organizado el presente documento es la siguiente: en el primer capítulo se presentan algunos temas de programación avanzada en TeX y en LaTeX; se hace particular énfasis en los mecanismos para definición de comandos y de entornos y en los diferentes tipos de estructuras condicionales implementados en TeX. En el segundo capítulo se presenta inicialmente la versión actual del paquete `background`, se proponen las mejoras (algunas sugeridas por la comunidad de usuarios y otras concebidas por el autor), se discute la estrategia de implementación de las mismas y finalmente se presenta el código comentado de la versión mejorada del paquete; se presentan también ejemplos comparativos de uso de las versiones antigua y nueva del paquete. En el tercer capítulo se sigue un esquema similar, para el caso del paquete `fancypar`. Finalmente, en los apéndices se incluye el código de las nuevas versiones de los dos paquetes.

1. Elementos de programación avanzada en TEX y en LATEX

> *Think of LATEX as a house built with the lumber and nails provided by TEX. You don't need lumber and nails to live in a house, but they are handy for adding an extra room. Most LATEX users never need to know any more about TEX commands than they can learn from this book. However, the lower-level TEX commands described in the* TEXbook *can be very useful when creating a new package for LATEX.*
>
> Leslie Lamport.
> LATEX: A Document Preparation System, 1 984

En este capítulo se presentan algunos elementos de programación en TEX y en LATEX. El tratamiento de ninguna manera es exhaustivo, ya que no es ese el principal objetivo del presente trabajo. El énfasis se hace en dos aspectos esenciales para la programación en lenguajes de macros (como TEX y LATEX): las definiciones y redefiniciones de macros y de entornos y el manejo de estructuras condicionales. Quien quiera adentrarse verdaderamente en las «entrañas» de TEX deberá estudiar alguno de los libros clásicos *TEXbook*, escrito por Donald E. Knuth [3] o el excelente libro *TEX by Topic*, de Viktor Eijkhout [2]. Para comprender a fondo cómo funciona LATEX, las referencias obligadas son el libro *The LATEX Companion, 2nd Edition* [5] y la versión documentada del kernel de LATEX, disponible en CTAN [6].

1.1. Definiciones simples en TEX y en LATEX

En esta sección se presentan las principales técnicas utilizadas para definir y redefinir macros y entornos. Se presentan también los métodos de creación y manipulación de contadores.

1.1.1. Definición de comandos en TEX y en LATEX

Un nuevo comando se define en LATEX utilizando `\newcommand`. Lo primero que hace el macro `\newcommand` es verificar que el comando que se va a definir no haya sido definido antes; si el comando ya había sido definido, el intento de volver a definirlo con `\newcommand` produce un error. La definición exacta de `\newcommand` es algo compleja y no es del caso considerarla aquí; nos

interesa eso sí mencionar que en últimas, \newcommand termina utilizando el comando primitivo \def. Cuando se programa en LaTeX, por norma general, debe usarse \newcommand para definir nuevos comandos, pero a veces resulta conveniente e incluso recomendable recurrir a la primitiva \def. Examinamos a continuación estos dos comandos.

Cuando se escribe un paquete o una clase, se deben tener en cuenta dos prácticas recomendables: (a) usar el caracter especial @ en los nombres de los comandos definidos por el paquete o la clase, con el fin de «esconderlos» del usuario, evitando así que éste accidentalmente los redefina y (b) utilizar un prefijo en la definición de los comandos internos; esto garantiza que los comandos definidos por el paquete o clase serán únicos y no causarán conflictos con comandos de otros paquetes, otras clases o del kernel de LaTeX. A manera de ejemplo, si el paquete se llama pac, entonces es una buena práctica llamar los comandos internos definidos por el paquete pac en la forma \pac@<nombre>, en donde <nombre> sigue las convenciones para los nombres válidos de un comando que se describen más adelante.

El comando primitivo que utiliza TeX para la definición de comandos es \def; su sintaxis es la siguiente:

```
\def\<nombre><parámetros>{<lista de tokens>}
```

en donde <nombre> es una cadena formada exclusivamente por letras mayúsculas y/o minúsculas y <parámetros> es la lista de los argumentos que va a tener el nuevo comando; esta lista puede ser vacía. Ejemplos de posibles definiciones son las siguientes:

```
\def\micoi{...}
\def\micoii#1#2#3{...}
\def\pac@mico#1#2{...}
```

En el primer caso, se define un comando sin argumentos \micoi; en el segundo caso hemos definido el comando \micoii con tres argumentos y, en último lugar, hemos definido un comando \pac@mico con dos argumentos; si esta definición aparece en un documento .tex, será necesario encerrarla entre \makeatletter, \makeatother debido a la presencia del caracter especial @.

Las definiciones anteriores pueden hacerse en LaTeX utilizando el comando \newcommand*, en cuyo caso deberíamos escribir

```
\newcommand*{\micoi}{...}
\newcommand*{\micoii}[3]{...}
\newcommand*{\pac@mico}[2]{...}
```

o incluso

```
\newcommand*\micoi{...}
\newcommand*\micoii[3]{...}
\newcommand*\pac@mico[2]{...}
```

ya que las llaves que agrupan \<nombre> no son realmente necesarias (\newcommand espera un solo token como primer argumento). Es de notar que hemos usado la variante \newcommand* que indica que nuestros nuevos comandos no admiten \par en la <lista de tokens>, que es exactamente lo que sucede al usar **\def**. Si queremos que nuestros comandos acepten \par, debemos definirlos usando **\newcommand** (sin el asterisco) o, **\long\def**.

Demos ahora una mirada detallada a la utilización del macro \newcommand y de su comando asociado \renewcommand, utilizado para redefinir comandos ya definidos. La sintaxis de estos dos comandos es (recuérdese que las llaves que delimitan el primer argumento no son necesarias):

```
\newcommand\<nombre>[<número>]{<texto>}
\renewcommand\<nombre>[<número>]{<texto>}
```

o, para el caso de un comando con un argumento opcional,

```
\newcommand\<nombre>[<número>][<val. defecto>]{<texto>}
\renewcommand\<nombre>[<número>][<val. defecto>]{<texto>}
```

en donde, en todos los casos, <número> ha de ser un entero entre cero y nueve; usar cero indica que el argumento opcional puede omitirse. Este entero indica el número de parámetros que se pueden usar en <texto> y también indica la cantidad de argumentos que LaTeX buscará cuando se invoque \<nombre>.

El identificador del comando, <nombre>, solo puede contener letras mayúsculas y/o minúsculas y *no* puede comenzar con la cadena end que está reservada en LaTeX para la definición del comando que controla el final de un entorno. Si el comando que se va a definir usando \newcommand ya había sido definido previamente o si el nombre del comando comienza con la cadena end, se emitirá un mensaje de error como el siguiente

```
! LaTeX Error: Command \<nombre> already defined.
               Or name \end... illegal, see p.192 of the manual.
```

Cuando se usa la segunda variante, se trata de una definición de un comando que admite un argumento opcional, cuyo valor normal está dado por <val. defecto>; este argumento opcional siempre será referido con el parámetro #1. Esta restricción sobre el número de argumentos opcionales es no esencial (ver la sección 1.1.1), pero la restricción sobre el máximo número total de argumentos sí lo es; existen técnicas que permiten definir macros con más de nueve argumentos, pero es preferible no usar este tipo de macros, a menos que realmente se sepa lo que se está haciendo.

Supongamos, a manera de ilustración, que queremos definir un comando \Mipot que escriba su argumento elevado a la quinta potencia, pero queremos eventualmente tener la posibilidad de cambiar el exponente. Podríamos entonces hacer la siguiente definición:

```
\newcommand\Mipot[2][5]{#2^{#1}}
```

Con esta definición, `$\Mipot{x}$` produce x^5, pero `$\Mipot[8]{x}$` produce x^8.

Es importante recordar que el máximo número de argumentos es siempre 9; por lo tanto, nuestro comando `\Mipot` podría tener máximo ocho argumentos obligatorios; también es importante saber que con `\renewcommand` se puede redefinir un comando usando cualquier número de parámetros, independientemente de cuántos tenía el comando al momento de su definición original.

Una precaución que debe tenerse es la de *no* definir un comando en términos de sí mismo, ya que esto produce un ciclo infinito. Una definición como la siguiente

```
\newcommand{\ciclo}{Ciclo\ciclo}
```

hará que al usar `\ciclo`, en el cuerpo de un documento, se produzca un mensaje de error del estilo:

```
! TeX capacity exceeded, sorry [main memory size=3000000].
\ciclo ->Ciclo
               \ciclo
l.7 \ciclo
```

y habrá que forzar la interrupción del proceso de compilación.

Comparación entre `\def` y `\newcommand`

- `\newcommand` verifica inicialmente si el comando que va a ser definido ya existe o no; `\def` no realiza esta verificación y permite sobreescribir un comando ya definido. Utilizando el comando `\@ifundefined` es posible realizar la verificación al emplear `\def`.

- `\newcommand` permite definir de manera directa comandos con un argumento opcional. La primitiva `\def` no permite inherentemente la definición de argumentos opcionales pero, tal como se muestra más adelante, es posible definir comandos con cualquier número de argumentos opcionales usando `\def`.

- A diferencia de `\newcommand`, `\def` permite la utilización de delimitadores distintos de las llaves y los corchetes.

Definición de comandos con dos o más argumentos opcionales

Como ya mencionamos, en principio solo es posible definir comandos con máximo un argumento opcional; sin embargo, en LATEX mismo existen comandos con más argumentos opcionales (`\parbox`, por ejemplo, admite tres comandos opcionales). ¿Cómo se pueden definir este tipo de comandos? En el siguiente ejemplo se muestra de manera esquemática el procedimiento para definir comandos con dos argumentos opcionales. La idea es utilizar `\@ifnextchar` para examinar el siguiente caracter en busca de un corchete de apertura que indique la presencia de los argumentos opcionales y definir comandos auxiliares que manipulen estos argumentos:

```
\def\micomando{%
```

```
  \@ifnextchar[%
    {\micomando@i}
    {\micomando@i[<val. defecto para opcional 1>]}%
}
\def\micomando@i[#1]{%
  \@ifnextchar[%
    {\micomando@ii{#1}}
    {\micomando@ii{#1}[<val. defecto para opcional 2>]}%
}
\def\micomando@ii#1[#2]#3{%
  <acciones>
}
```

Obviamente, el esquema anterior puede extenderse al caso de tres o más argumentos opcionales (nunca más de nueve, por supuesto). Es conveniente recordar que un comando definido usando el esquema anterior no acepta `\par`; si se desea que el comando acepte `\par`, será necesario usar `\long\def` en lugar de `\def`.

Otra opción, aplicable tan solo si se requieren comandos con *dos* argumentos opcionales, es utilizar el paquete `twoopt` [13] y sus comandos `\newcommandtwoopt`, `\renewcommandtwoopt` y `\providecommandtwoopt`. Por ejemplo,

```
\usepackage{twoopt}
\newcommandtwoopt{\<nombre>}[3][Def1][Def2]{<Acciones con #1, #2 y #3>}
```

En LaTeX3 la definición de comandos y entornos es mucho más flexible, aunque se mantiene la restricción de máximo nueve argumentos.

Definición de comandos con `\DeclareRobustCommand`

El comando `\DeclareRobustCommand` tiene la misma sintaxis que `\newcommand` y, en general, debe usarse solo cuando se está escribiendo un paquete o una clase. En el caso en que un token aparezca en un argumento móvil (por ejemplo, en el argumento de un comando seccional, o en el argumento de `\caption`), es necesario evitar la expansión del token. Esto ya que el argumento será escrito en un archivo auxiliar y la expansión del token al momento de la escritura puede no ser conveniente (de hecho, puede producir errores). Los comandos cuya expansión al momento de escritura produce error se denominan comandos *frágiles*. La solución tradicional para evitar los problemas que puede ocasionar la expansión de un comando frágil es «proteger» el comando, preveniendo la expansión mediante el comando `\protect`. En LaTeX 2_ε, el comando `\DeclareRobustCommand` puede ser utilizado (por los autores de paquetes o de clases) para definir comandos que son de una vez «robustos», es decir, que no son frágiles.

Como curiosidad histórica, los comandos `\(` y `\)` (que en LaTeX reemplazan a los comandos

TEX para modo matemático $ en línea) *no* fueron definidos como comandos robustos en el kernel, lo cual puede ocasionar errores cuya causa resulta difícil de comprender incluso para usuarios de cierto nivel. A manera de ejemplo, considérese el siguiente documento sencillo:

```
\documentclass{article}

\begin{document}

\tableofcontents
\section{La ecuaci\'on \( E = mc^2 \)}

\end{document}
```

cuando se compila este documento por segunda vez (es decir, cuando se va a generar el índice general), se obtiene el oscuro mensaje de error siguiente:

```
! LaTeX Error: Bad math environment delimiter.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.1 ...ak or <return> to continue without it.}}{1}

?
```

La causa del error es precisamente que se han usado comandos frágiles en el argumento obligatorio de un comando seccional y este argumento es móvil. Una manera de evitar el error es protegiendo estos comandos frágiles:

```
\documentclass{article}

\begin{document}

\tableofcontents
\section{La ecuaci\'on \protect\( E = mc^2 \protect\)}

\end{document}
```

Claramente esto puede resultar tremendamente engorroso. Afortunadamente existe una solución mucho más sencilla: cargar el paquete `fixltx2e` que, entre otras cosas, se encarga de proteger `\(`, `\)` y otros comandos que por descuido fueron dejados como frágiles en el kernel.

Definición de comandos con \providecommand

El comando \providecommand tiene también la misma sintaxis que \newcommand, pero no se encarga de verificar si el comando que va a ser definido ya existía previamente, lo cual puede ser causa de errores indeseados. A manera de ilustración, consideremos las siguientes definiciones del comando \end:

```
\newcommand{\end}{Endomorfismo}
```

al compilar el documento, incluso sin usar el comando \end, se produce un error, debido al intento de definir mediante \newcommand un comando cuyo nombre comienza con la cadena end. Por otro lado, si usamos

```
\providecommand{\end}{Endomorfismo}
```

entonces no se producirá ningún error mientras \end no sea usado en el documento. Al usar \end, definido como antes mediante \providecommand, se producirá un error que arroja un mensaje un tanto críptico:

```
! TeX capacity exceeded, sorry [input stack size=5000].
\end #1->\csname end#1
                       \endcsname \@checkend {#1}\expandafter \endgroup \
                          if@e...
l.6 \end
        {document}
```

La moraleja es clara: a menos que realmente se sepa lo que se está haciendo, es preferible usar \newcommand en lugar de \providecommand

1.1.2. Definición de entornos en LATEX

El comando que permite definir nuevos entornos en LATEX es \newenvironment. La sintaxis de este comando y de su similar \renewenvironment utilizado para redefinir un entorno ya existente, es similar a la de los comandos correspondientes para definir o redefinir comandos; sin embargo, hay una diferencia fundamental: el nombre del entorno *no* debe contener la diagonal invertida que debe estar presente en la definición o redefinición de un comando:

```
\newenvironment{<nombre>}[<número>]{<inicio>}{<fin>}
\renewenvironment{<nombre>}[<número>]{<inicio>}{<fin>}
```

o, para el caso de entornos con un argumento opcional,

```
\newenvironment{<nombre>}[<número>][<val. defecto>]{<inicio>}{<fin>}
\renewenvironment{<nombre>}[<número>][<val. defecto>]{<inicio>}{<fin>}
```

El <nombre> de un entorno puede estar formado por casi cualquier caracter; el asterisco (*) usualmente se emplea para indicar una variante de otro entorno que tiene el mismo nombre, pero

sin el asterisco. Aunque hay bastante libertad para escoger el nombre de un nuevo entorno, es recomendable utilizar solo caracteres alfabéticos y, posiblemente, el asterisco (después de todo, un entorno llamado +@123! no tiene mucho sentido).

Por cuestiones de mayor legilibilidad en el código, es recomendable escribir la definición de un nuevo entorno en la forma

```
\newenvironment{<nombre>}[<número>]
  {<inicio>}
  {<fin>}
```

Revisando la definición de \@newenv, que es la versión interna del comando \newenvironment, tal como está dada en el kernel de LaTeX [6, pág. 27 y 28], encontramos:

```
\long\def\@newenv#1#2#3#4{%
  \@ifundefined{#1}%
    {\expandafter\let\csname#1\expandafter\endcsname
            \csname end#1\endcsname}%
    \relax
  \expandafter\new@command
    \csname #1\endcsname#2{#3}%
<autoload>\aut@global
    \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}
```

Es decir, cuando se define un nuevo entorno utilizando

```
\newenvironment{<nombre>}[<número>]{<inicio>}{<fin>}
```

lo que realmente se está haciendo es una doble definición del estilo siguiente:

```
\newcommand\<nombre>[<número>]{<inicio>}
\def\end<nombre>{<fin>}
```

y esto tiene varias consecuencias:

- Como en la primera parte se usa \newcommand, entonces el intentar definir un entorno que ya ha sido definido producirá un error.
- Los parámetros declarados en la definición del entorno están disponibles en <inicio>, pero no en <fin>; esto explica el porqué en <fin> no se puede hacer referencia directa a los parámetros que se han declarado para el entorno que va a ser creado. Esta situación puede, sin embargo, remediarse guardando los parámetros, por ejemplo, en macros en la parte <inicio> y luego usando estos macros en la parte <fin>.

A manera de ilustración, supongamos que queremos crear un nuevo entorno que se comporte como el entorno estándar quote para citas, pero que reciba un argumento opcional para escribir el autor de la cita. El intento obvio

```
\newenvironment{citaautor}[1][]
  {\begin{quote}}
  {\par\hfill#1\end{quote}}
```

fracasará (pues se está usando indebidamente un parámetro en <fin>) con un mensaje de error como el siguiente:

```
! Illegal parameter number in definition of \endcitaautor.
<to be read again>
                    1
l.8   {\par\hfill#1\end{quote}}
```

La forma correcta de proceder, como ya se mencionó, es almacenar el parámetro en un macro y luego usar el macro en <fin>:

```
\newenvironment{citaautor}[1][]
  {\def\autorc{#1}\begin{quote}}
  {\par\hfill\autorc\end{quote}}
```

Para finalizar esta sección, mencionemos que existe un comando \provideenvironment que también permite definir entornos; el uso de este comando está sujeto a las mismas restricciones que su homólogo \providecommand para la definición de comandos.

1.1.3. Definición y manipulación de contadores en LATEX

Para definir un nuevo contador en LATEX se utiliza el comando \newcounter, cuya sintaxis es la siguiente:

```
\newcounter{<contador1>}[<contador2>]
```

en donde <contador1> es el nombre del contador que se va a definir. Las definiciones hechas mediante \newcounter son globales e inicializan el contador definido en cero. Si el nombre usado para el contador es igual al nombre de un contador ya existente, se produce un error. Por ejemplo, si escribimos

```
\newcounter{page}
```

obtendremos un mensaje de error como el siguiente:

```
! LaTeX Error: Command \c@page already defined.
               Or name \end... illegal, see p.192 of the manual.
```

Al usar \newcounter{<contador>} se crean internamente los dos comandos \c@<contador> y \the<contador>; el primero es el registro interno usado para el contador y el segundo controla la *representación* de ese registro (es decir, el símbolo que será impreso en el documento como representación del registro interno).

El argumento opcional se utiliza para subordinar el nuevo contador a un contador <contador2> ya existente. Esto quiere decir que el nuevo contador se reinicializa cada vez que <contador2> se incremente con \stepcounter o con \refstepcounter y adicionalmente, hace que el comando \the<contador1> se expanda a \arabic{<contador1>}.

Manipulación de contadores

Existen varios comandos que permiten modificar el valor de un contador ya definido. Las instrucciones

```
\setcounter{<contador>}{<valor>}
\addtocounter{<contador>}{<valor>}
```

se encargan de asignar y de sumar, respectivamente, el entero <valor> al contador <contador>. Estas modificaciones tienen efecto global sobre el contador.

Las instrucciones

```
\stepcounter{<contador>}
\refstepcounter{<contador>}
```

se encargan de incrementar globalmente en una unidad el valor de <contador> y de reinicializar todos los contadores subsidiarios de <contador>. El comando \refstepcounter adicionalmente define el valor \ref en uso como el valor generado por \the<contador>.

Es importante tener en cuenta que incrementar un contador es una operación global, pero la asignación del valor \ref en uso es una operación local.

Existen además los comandos \value, \alph, \Alph, \arabic, \roman, \Roman y \fnsymbol; todos ellos tienen como argumento obligatorio un contador ya existente. El comando \value produce el valor en uso de un contador; \arabic hace que el contador sea representado como un numeral arábigo, \roman y \Roman hacen que el contador sea representado usando numerales romanos en minúscula y en mayúscula, respectivamente.

Los comandos \alph y \Alph producen la representación del contador utilizando caracteres alfabéticos en minúscula y en mayúscula, respectivamente. Esto impone como restricción obvia que el valor del contador esté entre 0 y 26 inclusive; en caso contrario, se producirá un error. El comando \fnsymbol representa el contador como un símbolo de nota al pie de página; en este caso, el valor del contador debe estar entre 0 y 9 para evitar errores (si se usa algún paquete, como por ejemplo footmisc, el rango para el valor del contador se extiende).

Finalmente, \the<contador> permite obtener la representación visual de <contador>. Como ejemplo, en el archivo de clase article.cls encontramos las siguientes definiciones para los contadores que controlan las unidades seccionales:

```
\newcounter{part}
\newcounter{section}
```

```
\newcounter{subsection}[section]
\newcounter{subsubsection}[subsection]
\newcounter{paragraph}[subsubsection]
\newcounter{subparagraph}[paragraph]
\renewcommand\thepart{\@Roman\c@part}
\renewcommand\thesection{\@arabic\c@section}
\renewcommand\thesubsection{\thesection.\@arabic\c@subsection}
\renewcommand\thesubsubsection{\thesubsection.\@arabic\c@subsubsection}
\renewcommand\theparagraph{\thesubsubsection.\@arabic\c@paragraph}
\renewcommand\thesubparagraph{\theparagraph.\@arabic\c@subparagraph}
```

en donde `\@arabic` y `\@Roman` son las versiones internas (del kernel de LaTeX) de `\arabic` y `\Roman`, respectivamente.

1.2. Definición simultánea de un comando y un entorno

El material de tipo `verbatim` creado, por ejemplo, con `\verb` no puede aparecer como argumento de otros comandos, pero sí puede aparecer en el cuerpo de un entorno. Por esa razón resulta a veces conveniente disponer de un entorno y de un comando con el mismo nombre. Por supuesto, cada uno de ellos podría definirse por separado, pero esto implicaría una redundancia que es preferible evitar. En esta sección discutimos una técnica que permite evitar tal redundancia al posibilitar la definición simultánea de un entorno y un comando que compartan el nombre.

En la sección 1.1.2 vimos cómo internamente actúa `\newenvironment` definiendo en realidad dos nuevos comandos; este hecho permite usar una técnica harto útil: la definición simultánea de un comando y un entorno. La idea es definir un nuevo entorno y verificar (usando `\@ifnextchar`) si el siguiente caracter después del nombre del entorno es una llave de apertura; si este es el caso, se procede a la definición del comando y, en caso contrario, se da la definición para el caso del entorno propiamente dicho.

La definición que se hace, en caso de tratarse del comando se realiza usando un registro temporal cuyo argumento guarda el contenido del comando hasta que se cierra el entorno para luego ser usado, tal como ilustra el siguiente esquema:

```
\newenvironment{<nombre>}[i][]{%
  \begingroup
  <comandos>
  % Verificar si se usa como entorno o como comando:
  \@ifnextchar\bgroup
    % Si se usa como comando:
    {\long\def\@tempa####1{\vbox\bgroup####1\end<nombre>}\@tempa}
    % Si se usa como entorno
```

```
    {\vbox\bgroup}%
}
{
  <comandos>
  \egroup%
  \endgroup
}%
```

Ejemplo 1.2.1

A manera de ejemplo sencillo, definamos simultáneamente un comando y un entorno que se encarguen de escribir el contenido en color rojo:

```
\documentclass{article}
\usepackage{xcolor}

\makeatletter
\newenvironment{MiComEnt}[1][]{%
  \begingroup
  \color{red}
  \@ifnextchar\bgroup
    {\long\def\@tempa####1{\vbox\bgroup####1\endMiComEnt}\@tempa}
    {\vbox\bgroup}%
}
{
  \egroup%
  \endgroup
}%
\makeatother

\begin{document}

\MiComEnt{texto simple de prueba}

\begin{MiComEnt}
texto de prueba con material \verb!verbatim!.
\end{MiComEnt}

\end{document}
```

☑

1.3. Condicionales

En la construcción de macros realmente potentes, los condicionales son una herramienta fundamental. TEX ofrece una amplia gama de condicionales. En esta sección se listan todos ellos y se hace una descripción detallada de los principales. Algunos paquetes LATEX ofrecen otros comandos para facilitar el manejo de condicionales; el lector interesado puede consultar la documentación de los paquetes `ifthen` y `etoolbox`, por ejemplo.

1.3.1. Forma que toman los condicionales en TEX

Todo condicional en TEX es de una de las dos formas siguientes:

```
\if...<tokens de condición>
  <acciones si condición verdadera>
\fi
```

o

```
\if...<tokens de condición>
  <acciones si condición verdadera>
\else
  <acciones si condición falsa>
\fi
```

en donde <`tokens de condición`> son cero o más tokens que dependen del condicional en particular, <`acciones si condición verdadera`> es una serie de tokens que se procesarán si la evaluación resulta verdadera y <`acciones si condición falsa`> es una serie de tokens que se procesarán si la evaluación resulta falsa; cualquiera de estas últimas dos, o las dos, pueden ser vacías.

1.3.2. Algunos de los condicionales

En el cuadro 1.1 se muestran todos los comandos relacionados con el manejo de condicionales en TEX y se da una breve descripción de cada uno de ellos. En esta sección examinamos con mayor detalle algunas de estas estructuras condicionales.

Comparaciones con `\if`

La igualdad de códigos de caracter puede evaluarse usando

```
\if<token1><token2>
```

Para que los comandos puedan ser considerados como tokens, TEX les asigna el código de caracter 256 (el mínimo entero positivo que no es código de caracter de un token de caracter). Por lo tanto, todos los comandos son iguales al ser evaluados por `\if` y un comando es diferente de cualquier token de caracter.

Cuadro 1.1. Comandos TEX asociados a condicionales

Comando	Descripción
`\if`	Evaluar igualdad de de códigos de caracter.
`\ifcat`	Evaluar igualdad de de códigos de categoría.
`\ifx`	Evaluar igualdad de de códigos de caracter de expansión de macros o igualdad de código de caracter y código de categoría.
`\ifcase`	Expresión por casos.
`\ifnum`	Evaluar relaciones entre números.
`\ifodd`	Evaluar si un número es impar.
`\ifhmode`	Evaluar si el modo en uso es modo horizontal (posiblemente restringido).
`\ifvmode`	Evaluar si el modo en uso es modo vertical (posiblemente interno).
`\ifmmode`	Evaluar si el modo en uso es modo matemático (posiblemente desplegado).
`\ifinner`	Evaluar si el modo en uso es interno.
`\ifdim`	Comparar dos dimensiones.
`\ifvoid`	Evaluar si un registro de caja está vacío.
`\ifhbox`	Evaluar si un registro de caja contiene una caja horizontal.
`\ifvbox`	Evaluar si un registro de caja contiene una caja vertical.
`\ifeof`	Evaluar el fin de la entrada o la no existencia del archivo.
`\iftrue`	Arroja siempre `true`.
`\iffalse`	Arroja siempre `false`.
`\fi`	Delimitador que cierra todos los condicionales.
`\else`	Escoge `<acciones si condición falsa>` en un condicional o el caso por defecto en un `\ifcase`.
`\or`	Separador para las entradas en un `\ifcase`.
`\newif`	Crea un nuevo condicional.

Ejemplo 1.3.1

Supongamos que realizamos las asignaciones siguientes:

```
1 \catcode`\z=13
2 \catcode`\y=13
3 \def z{r}
4 \def y{r}
```

Entonces, `\if zy` es verdadero, pues tanto `b` como `c` (que se han vuelto activos) se expanden al caracter a mientras que `\if\noexpand z\noexpand y` es evidentemente falso. ☑

Comparaciones con `\ifcat`

Los códigos de categoría son ignorados por `\if`. Para evaluar igualdad entre códigos de categoría, se puede usar `\ifcat`:

```
\ifcat<token1><token2>
```

A los comandos se les asigna código de categoría 16, con lo cual todos los comandos son iguales entre sí y diferentes de los tokens de caracter.

Ejemplo 1.3.2

Suponiendo los códigos de categoría definidos como en el formato Plain, en un documento `.tex`

```
\ifcat a@
```

es falso, pero

```
\makeatletter
\ifcat a@
\makeatother
```

es verdadero pues `\makeatletter` cambia el código de categoría del caracter `@` de `12` (`<<other>>`)a `11` (`<<letter>>`). ☑

Comparaciones numéricas

La evaluación de relaciones entre números puede realizarse con

```
\ifnum<número1><relación><número1>
```

en donde `<relación>` es uno de los caracteres <, =, or > (de categoría 12). Para evaluar la paridad de un número, puede usarse `\ifodd`. La expresión

```
\ifodd<número>
```

es verdadera si `<número>` es impar y es falsa en caso contrario.

Comparaciones entre dimensiones

Las relaciones entre dimensiones pueden evaluarse utilizando

```
\ifdim<dimensión1><relación><dimensión2>
```

utilizando como `<relación>` las mismas tres relaciones que para `\ifnum`.

Ejemplo 1.3.3

A manera de ilustración, el kernel de LaTeX contiene la siguiente asignación que busca evitar problemas con `\twocolumn` cuando `\topskip` tenga un valor de `0pt`:

```
\ifdim\topskip<1sp\global\topskip 1sp\relax\fi
```

☑

Expresiones por casos

TEX dispone de una expresión por casos llamada `\ifcase` y tiene la siguiente sintaxis:

```
\ifcase<número><caso_0>\or...\or<caso_n>\else<otros casos>\fi
```

en donde hay $n-1$ comandos `\or` para n casos. Cada una de las partes `<caso_i>` puede ser vacía y la parte `\else<otros casos>` es opcional.

Ejemplo 1.3.4

A manera de ilustración, el comando interno `\@alph` que se encarga de representar un contador utilizando letras minúsculas del alfabeto latino, está definido en el kernel de la siguiente manera:

```
\def\@alph#1{%
\ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
  k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or
    y\or z\else\@ctrerr\fi}
```

en donde `\@ctrerr` es el comando del kernel [6, pág. 43] que se invoca cuando se supera el límite de 26 para la representación alfabética de un contador, y se encarga de producir el siguiente mensaje de error:

```
! LaTeX Error: Counter too large.
```

☑

Definición de nuevos condicionales

El macro `\newif` permite definir nuevos condicionales. La expresión

```
\newif\iffoo
```

hace que TEX cree tres nuevos comandos `\footrue`, `\foofalse` e `\iffoo`; los dos primeros permiten asignar el valor `true` o `false`, respectivamente, a la condición y el tercero evalua la condición `foo`.

Ejemplo 1.3.5

El kernel de LATEX define un «switch» booleano de uso genérico `\newif\if@tempswa` que es usado, por ejemplo en la definición de `\cite`, para determinar la forma de las citas en el texto, dependiendo de si se usa o no el argumento opcional del comando:

```
\DeclareRobustCommand\cite{%
  \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}
```

☑

Condicionales del tipo `si {<cond1> o <cond2>} entonces`

El macro `\or` solo puede utilizarse dentro de condicionales de tipo `\ifcase`, así que la implementación, por ejemplo, de un condicional `\ifnum` en que la condición de control sea de tipo compuesto {<cond1> o <cond2>} no puede hacerse utilizando el macro `\or`. Para implementar un condicional de esta clase existen varias alternativas: una de ellas es utilizar condicionales «auxiliares» para convertir el problema original en uno simple de tipo `true/false`:

```
\ifnum\ifnum\x=1 1\else\ifnum\x=2 1\else0\fi\fi
   =1 %
  <acciones en caso de ser 1 o 2>
 \else
  <acciones en caso contrario>
 \fi
```

El macro `\ifnum` «externo» sencillamente evalúa una situación simple de tipo 0 o 1.

1.4. Cómo realizar copias de comandos

El comando primitivo `\let` asigna el significado actual de un token a un comando o a un caracter activo. Es decir, se puede utilizar (y, de hecho, es una práctica muy usual hacerlo) `\let` para definir un comando como sinónimo de un token. La sintaxis del comando `\let` es la siguiente:

`\let\<nombre>=token`

en donde el signo de igualdad es opcional. Una primera ilustración de esta asignación de sinónimos queda ejemplificada por la definición que PlainTEX hace de `\bgroup` y `\egroup`:

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

Esta definición de sinónimos resulta una herramienta bastante útil, ya que permite redefinir un comando manteniendo la funcionalidad de su significado original. Por ejemplo, si queremos redefinir el entorno `quote` para que el texto aparezca en itálica, podríamos hacer lo siguiente:

```
\let\oldquote=\quote
\let\oldendquote=\endquote
\renewenvironment{quote}
   {\oldquote\itshape}
   {\oldendquote}
```

Con estas definiciones, tenemos ahora dos entornos: `quote` y `oldquote`; el primero escribe el contenido en itálica, en tanto que el segundo se comporta como el entorno original `quote`.

Hay, sin embargo, una restricción importante al uso de `\let`: es preferible *no* utilizarlo (a menos que realmente se sepa lo que se está haciendo) para crear sinónimos de comandos que reciban un

argumento opcional. Para ilustrar los problemas que podrían presentarse al ignorar esta advertencia, consideremos un ejemplo sencillo: cuando escribimos

```
\newcommand\mico[2][A]{-#1-#2-}
```

la de finición real de \mico es

```
\@protected@testopt \mico \\mico {A}
```

El comando \@protected@testopt verifica si se está en modo normal de escritura o si se está en alguna «situación especial» (por ejemplo, aquella en que deben procesarse de alguna manera los argumentos para escribirlos en archivos auxiliares). En el segundo caso, la acción que se realiza es simple: elimina todo y deja solamente \protect\mico (lo cual, según ya vimos, resulta bastante conveniente).

En el primer caso, se examina el siguiente caracter para determinar si se ha especificado o no el argumento opcional. Sin entrar en todos los detalles, la situación es la siguiente:

Si se usa \mico{MICO}, el resultado es \\mico[A]{MICO}. Si se usa \mico[NJDP]{MICO}, el resultado es \\mico[NJDP]{MICO}.

En los dos casos es importante notar que el comando que actúa es \\mico, ¡con una diagonal invertida como parte del nombre!. Ahora bien, ¿cuál es la definición de \\mico? Esto es lo que TEX dice:

```
> \\mico=\long macro:
[#1]#2->-#1-#2-.
```

El primer argumento es precisamente lo que está entre corchetes. Supongamos ahora que sacamos una copia de \mico para redefinirlo, de la manera siguiente:

```
\let\oldmico\mico
\renewcommand{\mico}[2][U]{\oldmico[#1]{#2}}
```

y que \mico[T]{MICO} aparece en modo de escritura normal. El resultado del procesamiento interno será, a grandes rasgos, el siguiente:

```
\mico[T]{MICO}
\\mico[T]{MICO}
\oldmico[T]{MICO}
\\mico[T]{MICO}
\oldmico[T]{MICO}
\\mico[T]{MICO}
...
```

y TEX entra en un bucle infinito. Esto se debe a que con el uso de \renewcommand se le ha dado un significado a \\mico y, como es fácil verificar, este significado es

```
> \\mico=\long macro:
[#1]#2->\oldmico[#1]{#2}
```

en tanto que el significado de `\oldmico` es exactamente el mismo significado del comando original `\mico` que encontrará al `\\mico` que tiene la nueva definición.

Para evitar este inconveniente, se puede usar el comando `\LetLtxMacro` implementado por el paquete `letltxmacro` [12]; al escribir

```
\usepackage{letltxmacro}
\LetLtxMacro\oldmico{\mico}
```

se hace, además de `\let\oldmico\mico`, la asignación `\let\\oldmico\\mico` (para los nombres que incluyen la diagonal invertida) y se cambia el significado de `\oldmico` de tal forma que ahora se expande a

```
\@protected@textopt \oldmico \\oldmico {A}
```

evitando así el ciclo infinito.

2. El paquete background

Este capítulo trata del paquete LaTeX 2ε `background`. En la primera sección se describe brevemente el paquete y se relacionan, de manera suscinta, otros paquetes actualmente disponibles en CTAN y que brindan una funcionalidad similar. En la sección 2.2 se presenta el código comentado de la versión actual del paquete y se muestran algunos ejemplos de uso. En la sección 2.3 se presentan los cambios que han sido sugeridos para mejorar la implementación del paquete. Finalmente, en la sección 2.4 se discuten las estrategias usadas para implementar los cambios sugeridos, se presenta la versión comentada del código de la nueva versión del paquete y se ilustra, mediante ejemplos, el uso de la nueva y mejorada interfaz.

2.1. Descripción del paquete

Este paquete permite poner material arbitrario de fondo (al estilo marca de agua) en un documento LaTeX 2ε. Existen en CTAN otros paquetes con una funcionalidad similar: el paquete `watermark` [16], implementado por Alexander I. Rozhenko, permite definir marcas de agua para páginas pares e impares y para una única página. El paquete `draftcopy` [19], implementado por Jürgen Vollmer, permite colocar «DRAFT» (u otro texto) en color gris claro diagonalmente o en la parte inferior de todas o algunas páginas del documento; este paquete utiliza el comando `\special`, así que no es posible utilizar directamente con pdflatex. El paquete `draftwatermark` [9], implementado por Sergio Callegari, permite agregar una marca de agua de tipo texto en todas las páginas o en la primera página de un documento; la marca de agua aparece diagonalmente en la(s) página(s).

El paquete `background` incorpora y amplía las posibilidades de los anteriores paquetes: permite marcas de agua de tipo texto y de tipo imágenes (externas e incluidas con el comando estándar `\includegraphics` del paquete `graphicx`, o creadas, por ejemplo, con `PGF/TikZ` [18]). El usuario tiene completo control sobre el contenido, el color, el tamaño, la posición y la opacidad del material de fondo. El paquete permite suprimir (o agregar) el material en alguna(s) página(s) particular(es) del documento. Ya que se apoya en TikZ, el documento puede compilarse con latex o con pdflatex. El usuario puede definir fácilmente diferentes materiales para las páginas pares o impares del documento y puede cambiar a voluntad el material para ciertas secciones del documento.

2.1.1. Información general

- El paquete se encuentra en CTAN (bajo la licencia LPPL) desde noviembre de 2009: `http://www.ctan.org/pkg/background`. En este momento se encuentra disponible la versión 1.0.
- El paquete se distribuye actualmente con MiKTeX y TeX Live (incluyendo, en particular, las distribuciones proTeXt y MacTeX).
- El paquete aparece reseñado como «paquete de especial interés» por los editores de la revista TUGBOAT [1].
- El paquete se encuentra registrado bajo el nombre «The Background Package» ante la Oficina de Registro de la Dirección Nacional de Derechos de Autor del Ministerio del Interior y de Justicia, bajo el radicado 13-25-209 del 8 de abril de 2010. Los derechos patrimoniales del paquete fueron cedidos a la Universidad Nacional de Colombia.
- El paquete incluye los archivos de código `background.dtx` y `background.ins`, el archivo de texto `README.txt`, el archivo generado `background.sty` y el archivo `background.pdf` (la documentación del paquete).

2.2. Implementación actual del paquete

En esta sección se presenta el código comentado de la actual versión del paquete. Inicialmente, se hace la introducción estándar, anunciando que se trata de un paquete LaTeX 2_ε y se cargan los paquetes requeridos:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{background}[2009/11/07 v1.0 bg material]

\RequirePackage{everypage}
\RequirePackage{tikz}
\RequirePackage{afterpage}
```

A continuación se declara una variable booleana `bg@some` que controla si el material debe o no aparecer en todas las páginas del documento, mediante las opciones de paquete `all` o `some`. Se inicializa esta variable en `false`, indicando que, por defecto, el material aparecerá en todas las páginas:

```
\newif\ifbg@some
  \bg@somefalse
```

Cuadro 2.1. Comandos de usuario del paquete background que permiten modificar los atributos del material de fondo

Comando	Descripción	Valor por defecto
\SetBgContents	El tipo de material que será presentado. Puede ser texto (arbitrario) o material gráfico externo o material gráfico creado, por ejemplo con PGF/TikZ.	Draft
\SetBgColor	El color del material de fondo.	red!45
\SetBgAngle	El ángulo del material de fondo.	60
\SetBgOpacity	El factor de opacidad que se aplicará al color del material de fondo.	0.5
\SetBgScale	El factor de escala que se aplicará al material de fondo.	15
\SetBgPosition	La posición del material de fondo.	current page.center
\SetBgAnchor	El punto de anclaje que será usado para el nodo que contiene el material de fondo.	—
\SetBgHshift	El desplazamiento horizontal que será aplicado al nodo que contiene el material de fondo.	0
\SetBgVshift	El desplazamiento vertical que será aplicado al nodo que contiene el material de fondo.	0

Se definen a continuación las variables internas que se utilizarán para controlar los atributos del material de fondo y se asignan los valores por defecto. La lista de los comandos de usuario que controlan estos atributos, junto con su descripción y los valores por defecto aparece en el cuadro 2.1.

```
\def\bg@contents{Draft}
\def\bg@color{red!45}
\def\bg@angle{60}
\def\bg@opacity{.5}
\def\bg@scale{15}
\def\bg@position{current page.center}
\def\bg@anchor{}
\def\bg@hshift{0}
\def\bg@vshift{0}
```

Declaramos ahora las opciones de paquete all, some, center. bottom y top y hacemos las redefiniciones de macros internos que controlan los atributos de cada una de las opciones. En el cuadro 2.2 se muestran estas opciones y su efecto.

```
\DeclareOption{all}{\bg@somefalse}
\DeclareOption{some}{\bg@sometrue}
\DeclareOption{center}{%
  \def\bg@position{current page.center}%
  \def\bg@anchor{}%
```

Cuadro 2.2. Las diferentes opciones del paquete `background` en su actual versión

Opción	Descripción
`all`	El material de fondo aparecerá en todas las páginas del documento.
`some`	El material de fondo aparecerá solo en aquellas páginas en que se use el comando `\BgThispage`.
`center`	El material de fondo aparecerá diagonalmente.
`bottom`	El material de fondo aparecerá horizontalmente en la parte inferior de la(s) página(s).
`top`	El material de fondo aparecerá horizontalmente en la parte superior de la(s) página(s).

```
  \def\bg@angle{60}}
\DeclareOption{bottom}{%
  \def\bg@position{current page.south}%
  \def\bg@anchor{above}%
  \def\bg@angle{0}%
  \def\bg@scale{8}}
\DeclareOption{top}{%
  \def\bg@position{current page.north}%
  \def\bg@anchor{below}%
  \def\bg@angle{0}%
  \def\bg@scale{8}}
```

Ahora hacemos que inicialmente se usen las opciones `all` y `center` y procesamos las opciones ya definidas.

```
\ExecuteOptions{all,center}
\ProcessOptions
```

Creamos ahora los comandos de usuario, que permitirán el control de los atributos.

```
\newcommand*\SetBgContents[1]{%
  \def\bg@contents{#1}}
\newcommand*\SetBgColor[1]{%
  \def\bg@color{#1}}
\newcommand*\SetBgAngle[1]{%
  \def\bg@angle{#1}}
\newcommand*\SetBgOpacity[1]{%
  \def\bg@opacity{#1}}
\newcommand*\SetBgScale[1]{%
  \def\bg@scale{#1}}
\newcommand*\SetBgPosition[1]{%
  \def\bg@position{#1}}
```

```
\newcommand*\SetBgAnchor[1]{%
  \def\bg@anchor{#1}}
\newcommand*\SetBgHshift[1]{%
  \def\bg@hshift{#1}}
\newcommand*\SetBgVshift[1]{%
  \def\bg@vshift{#1}}
```

LLegamos al comando central del paquete. Haciendo uso del entorno `tikzpicture` del paquete PGF/TikZ creamos un nodo que se encarga de mostrar el material de fondo; todos los atributos que han sido definidos (por defecto o por asignaciones del usuario a los comandos de atributo) se pasan a este nodo. El contenido del entorno `tikzpicture` se guarda en el comando `\bg@material`.

```
\newcommand\bg@material{%
  \begin{tikzpicture}[remember picture,overlay]
  \node [rotate=\bg@angle,scale=\bg@scale,opacity=\bg@opacity,%
    xshift=\bg@hshift,yshift=\bg@vshift,color=\bg@color]
    at (\bg@position) [\bg@anchor]  {\bg@contents};
  \end{tikzpicture}}%
```

El comando `\BgThispage` hace uso del comando `\AddThispageHook` implementado por el paquete `everypage` para sencillamente agregar el contenido de `\bg@material`.

```
\newcommand\BgThispage{\AddThispageHook{\bg@material}}
```

Con la ayuda del paquete `afterpage` definimos el comando de usuario `\NoBgThispage` que permite suprimir el material de fondo en aquellas páginas en que sea usado el comando. Este comando no debe utilizarse si la opción de clase `twoside` está activa o si se está en modo de dos o más columnas, debido a las limitaciones del comando `\afterpage`.

```
\newcommand\NoBgThispage{%
  \let\oldbg@material\bg@material\renewcommand\bg@material{}%
  \afterpage{\AddEverypageHook{\oldbg@material}}}
```

Dependiendo del valor de la variable booleana `bg@some` hacemos que se muestre el material de fondo (si la variable tiene el valor `false`) o que no se muestre nada (si la variable tiene el valor `true`).

```
\ifbg@some
  \AddThispageHook{}
\else
  \AddEverypageHook{\bg@material}
\fi
```

```
\endinput
```

2.2.1. Ejemplos de uso del paquete en su versión actual

En los ejemplos siguientes el paquete `lipsum` se utiliza exclusivamente para generar texto de manera automática.

Ejemplo 2.2.1

En el siguiente ejemplo se ilustra cómo usar el paquete `background` para producir un esquema personalizado para la numeración de páginas de un documento; en la figura 2.1 (pág. 37) se muestra el resultado obtenido:

```
\documentclass{article}
\usepackage[top]{background}
\usepackage{lipsum}

\SetBgContents{-\thepage-}
\SetBgAngle{0}
\SetBgColor{black!80}
\SetBgOpacity{1}
\SetBgScale{4}
\SetBgHshift{60}
\SetBgVshift{-5}

\pagestyle{empty}

\begin{document}
\lipsum[1-30]
\end{document}
```

☑

Ejemplo 2.2.2

En este ejemplo se muestra cómo proceder para obtener diferente material de fondo en las páginas pares e impares e un documento; en la figura 2.2 (pág. 37) se muestra el resultado obtenido:

```
\documentclass{article}
\usepackage{background}
\usepackage{lipsum}
\usepackage{ifthen}

\SetBgContents{}
\SetBgOpacity{1}
\SetBgScale{4}
\makeatletter
```

```
\AddEverypageHook{%
  \ifthenelse{\isodd{\value{page}}}%
    {\SetBgAngle{90}%
     \SetBgPosition{0,-.6\textheight}%
     \SetBgColor{blue}%
     \SetBgContents{The background package}}%
    {\SetBgAngle{270}%
     \SetBgPosition{1.35\textwidth,-.6\textheight}%
     \SetBgColor{red}%
     \SetBgContents{Gonzalo Medina}}%
  \bg@material}
\makeatother

\begin{document}
\lipsum[1-10]
\end{document}
```

☑

2.3. Cambios sugeridos

Durante el tiempo en que el paquete ha estado disponible al público general, como parte de las distribuciones MiKTeX y de TeX Live, se han concebido una serie de cambios en la implementación del paquete tendientes a mejorar su desempeño, a brindar nuevas opciones y a hacerlo más amigable para el usuario. Los cambios que serán incorporados en la nueva versión, cuyo diseño e implementación hacen parte de este trabajo, son los siguientes:

1. En la actual versión, no se realiza verificación alguna al momento de cargar internamente los paquetes requeridos por `background`. Aunque, en este caso en particular, esto no genera riesgo alguno (pues ninguno de los paquetes requeridos admite opciones de paquete), es preferible hacer la verificación inicial para evitar duplicaciones por parte del usuario al momento de cargar `background`.

2. En la actual versión, el control de atributos se hace mediante una serie de macros; es preferible hacer que las opciones de control de atributos se puedan pasar directamente como opciones de paquete, para hacer más amigable la interacción con el usuario y unificar la interfaz para el control de atributos y las opciones de paquete.

3. Aunque en el momento existe la posibilidad de cambiar los atributos del material de fondo (tantas veces como se desee dentro de un documento), es necesario que la personalización dinámica de las propiedades del material de fondo se pueda hacer de una manera más intuitiva

para el usuario, sin que este deba recordar y manipular los macros que actualmente controlan los atributos del material de fondo.

4. Evitar definitivamente al usuario el tener que hacer uso de comandos internos. En la actual versión, si se quiere hacer una definición de diferentes materiales de fondo para las páginas pares e impares del documento, el usuario tiene que manipular el comando interno `\bg@material` (ver el ejemplo 2.2.2) lo cual lleva al uso de `\makeatletter`, `\makeatother`. Sería preferible evitar esta situación, sobretodo para usuarios no experimentados.

5. Algunos usuarios han reportado un bicho («bug») que hace que la compilación de un documento de una sola página que use el paquete `background` con `pdflatex` produzca un archivo `PDF` defectuoso que no puede leerse. La solución con la actual versión es agregar manualmente `\newpage` o`\clearpage` al final del documento.

2.4. La nueva versión del paquete

En esta sección se discute la estrategia seguida para la implementación de los cambios sugeridos en la sección anterior; adicionalmente se presenta la versión comentada del código de la nueva versión del paquete. Finalmente, a manera de ilustración de las mejoras introducidas, se presentan nuevamente los ejemplos de la sección 2.2.1, pero utilizando la interfaz de la nueva implementación.

2.4.1. Estrategia de implementación de los cambios

1. Utilizando el paquete `xkeyval`, es posible lograr que las opciones que controlan los atributos del material de fondo se puedan pasar directamente como opciones de paquete. Esto trae las siguientes ventajas: (a) Brinda una interfaz más amigable para el usuario y (b) Permite unificar, bajo el sistema de <`clave`>=<`valor`>, el manejo de las antiguas opciones de paquete y de control de los atributos.

2. Pensando en obtener una forma más dinámica de cambiar los atributos del material de fondo (permitiendo estos cambios tanto en el preámbulo como tantas veces se desee dentro de un documento, de manera amigable), se decidió implementar un comando de usuario que, mediante el mecanismo <`clave`>=<`valor`>, permita modificar las propiedades del material de fondo; este comando podrá usarse tanto en el preámbulo como en el cuerpo del documento, tantas veces como se requiera.

3. Para evitar que el usuario deba manipular comandos internos, se han creado comandos de usuario, que no son otra cosa que copias de los comandos internos. En la nueva versión ya no es necesario utilizar `\makeatletter`, `\makeatother`. Comparar, como ilustración, el código de los ejemplos 2.2.2 (versión actual) y 2.4.2 (nueva interfaz).

4. Para evitar el bicho («bug») que hace que la compilación con `pdflatex` de un documento de una sola página que use la actual versión del paquete `background` produzca un archivo `PDF` defectuoso que no puede leerse, la nueva versión agrega automáticamente `\clearpage` al final del documento.

5. Se ha implementado un mecanismo que permite tener compatibilidad casi completa entre la versión actual del paquete y la nueva versión.

2.4.2. Código comentado de la nueva versión del paquete

Presentamos en esta sección el código comentado de la nueva versión del paquete. Los cambios introducidos se apoyan fuertemente en la utilización de algunos macros del «kernel» de LaTeX y de otros implementados por el paquete `xkeyval`.

Inicialmente, se hace la introducción estándar, anunciando que se trata de un paquete LaTeX 2_ε; adicionalmente, se cargan los paquetes requeridos. A diferencia de la anterior versión, la carga de los paquetes se realiza ahora solamente si no han sido cargados previamente por el usuario antes de requerir `background`; en esta nueva versión cargamos adicionalmente el paquete `xkeyval` para el manejo de las claves.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{background}[2011/03/27 v2.0 background material]

\@ifpackageloaded{xkeyval}{}{\RequirePackage{xkeyval}}
\@ifpackageloaded{tikz}{}{\RequirePackage{tikz}}
\@ifpackageloaded{everypar}{}{\RequirePackage{everypage}}
\@ifpackageloaded{afterpage}{}{\RequirePackage{afterpage}}
```

Declaramos ahora las opciones, con la ayuda del paquete `xkeyval`. Usamos `BG` como prefijo y `background` como identificador de la familia. Las opciones `contents`, `color`, `angle`, `opacity`, `scale`, `position`, `anchor`, `hshift` y `vshift` se encargan de definir sendos macros. La lista completa de las opciones disponibles, así como sus valores posibles y sus valores por defecto se presenta en el cuadro 2.3.

```
\DeclareOptionX[BG]<background>{contents}{%
  \def\BackgroundContents{#1}}
\DeclareOptionX[BG]<background>{color}{%
  \def\BackgroundColor{#1}}
\DeclareOptionX[BG]<background>{angle}{%
  \def\BackgroundAngle{#1}}
\DeclareOptionX[BG]<background>{opacity}{%
  \def\BackgroundOpacity{#1}}
\DeclareOptionX[BG]<background>{scale}{%
```

```
  \def\BackgroundScale{#1}}
\DeclareOptionX[BG]<background>{position}{%
  \def\BackgroundPosition{#1}}
\DeclareOptionX[BG]<background>{anchor}{%
  \def\BackgroundAnchor{#1}}
\DeclareOptionX[BG]<background>{hshift}{%
  \def\BackgroundHShift{#1}}
\DeclareOptionX[BG]<background>{vshift}{%
  \def\BackgroundVShift{#1}}
```

A continuación declaramos, al igual que en la versión anterior, una variable booleana `bg@some` que controla si el material debe o no aparecer en todas las páginas del documento, mediante las nuevas opciones de paquete `pages=all` o `pages=some`. Se inicializa esta variable en `false`, indicando que, por defecto, el material aparecerá en todas las páginas:

```
\newif\ifbg@some
  \bg@somefalse
```

Definimos ahora la clave de selección `pages`, con posibles valores `all` y `some`. Si el valor de `pages` es `all`, entonces se asigna `true` a la variable booleana `\bg@some`; si el valor de `pages` es `some`, entonces se asigna `false` a `\bg@some`. Si la entrada es inválida, se genera el mensaje «erroneous input ignored» (se ha ignorado la entrada errónea).

```
\define@choicekey+[BG]{background}{pages}[\val\nr]{all,some}{%
  \ifcase\nr\relax
    \bg@somefalse
  \or
    \bg@sometrue
  \fi
}{%
  \PackageWarning{background}{erroneous input ignored}%
}
```

Definimos ahora la clave de selección `placement`, con posibles valores `center`, `bottom` y `top`. Dependiendo de cada una de las valores válidos, se inicializan los macros de control de atributo apropiadamente. Si la entrada es inválida, se genera el mensaje «erroneous input ignored» (se ha ignorado la entrada errónea).

```
\define@choicekey+[BG]{background}{placement}[\val\nr]{center,bottom,top
    }{%
  \ifcase\nr\relax
    \renewcommand\BackgroundPosition{current page.center}%
    \renewcommand\BackgroundAnchor{}%
    \renewcommand\BackgroundAngle{60}
```

```
  \or
    \renewcommand\BackgroundPosition{current page.south}%
    \renewcommand\BackgroundAnchor{above}%
    \renewcommand\BackgroundAngle{0}%
  \or
    \renewcommand\BackgroundPosition{current page.north}%
    \renewcommand\BackgroundAnchor{below}%
    \renewcommand\BackgroundAngle{0}%
  \fi
}{%
  \PackageWarning{background}{erroneous input ignored}%
}
```

A continuación, utilizamos el macro `\DeclareOptionX*` con argumento `\CurrentOption` para producir un «warning» en el caso en que el usuario utilice una opción que no ha sido declarada.

```
\DeclareOptionX*{\PackageWarning{background}{'\CurrentOption' ignored}}
```

Ahora, usando `\ExecuteOptionsX`, asignamos valores por defecto a las claves que han sido credas usando `\DeclareOptionX` y luego procesamos los pares `<clave>=<valor>`.

```
\ExecuteOptionsX[BG]<background>{%
  contents=Draft,%
  color=red!45,%
  angle=60,%
  opacity=0.5,%
  scale=10,%
  position=current page.center,%
  anchor={},%
  hshift=0,%
  vshift=0%
}

\ProcessOptionsX[BG]<background>
```

Definimos ahora el comando de usuario `\backgroundsetup`. Este comando, se define como *robusto* y será el encargado de la modificación dinámica de los atributos del material de fondo. El comando recibe como argumento una lista de pares del estilo `<clave>=<valor>` y modifica el material de fondo según los valores asignados, utilizado `\setkeys`. El cambio a los atributos surtirá efecto desde el punto en que se invocó el comando, hasta el final del documento, o hasta que se realice un nuevo llamado a `\backgroundsetup`.

```
\DeclareRobustCommand*\backgroundsetup[1]{%
  \setkeys[BG]{background}{#1}
```

```
}
```

Como en la versión anterior, el comando `\bg@material` es el comando central del paquete. Haciendo uso del entorno `tikzpicture` del paquete `PGF/TikZ` creamos un nodo que se encarga de mostrar el material de fondo; todos los atributos que han sido definidos (por defecto o por asignaciones del usuario a los comandos de atributo) se pasan a este nodo. El contenido del entorno `tikzpicture` se guarda en el comando `\bg@material`.

```
\newcommand\bg@material{%
  \begin{tikzpicture}[remember picture,overlay,scale=\BackgroundScale]
  \node [rotate=\BackgroundAngle,scale=\BackgroundScale,
    opacity=\BackgroundOpacity,%
    xshift=\BackgroundHShift,yshift=\BackgroundVShift,color=\
       BackgroundColor]
    at (\BackgroundPosition) [\BackgroundAnchor]  {\BackgroundContents};
  \end{tikzpicture}}%
```

El comando `\BgThispage` hace uso del comando `\AddThispageHook` implementado por el paquete `everypage` para sencillamente agregar el contenido de `\bg@material`.

```
\newcommand\BgThispage{\AddThispageHook{\bg@material}}
```

Con la ayuda del paquete `afterpage` definimos el comando de usuario `\NoBgThispage` que permite suprimir el material de fondo en aquellas páginas en que sea usado el comando. Este comando no debe utilizarse si la opción de clase `twoside` está activa o si se está en modo de dos o más columnas, debido a las limitaciones del comando `\afterpage`.

```
\newcommand\NoBgThispage{%
  \let\oldbg@material\bg@material\renewcommand\bg@material{}%
  \afterpage{\AddEverypageHook{\oldbg@material}}}
```

Dependiendo del valor de la variable booleana `bg@some` hacemos que se muestre el material de fondo (si la variable tiene el valor `false`) o que no se muestre nada (si la variable tiene el valor `true`).

```
\ifbg@some
  \AddThispageHook{}
\else
  \AddEverypageHook{\bg@material}
\fi
```

Para tener compatibilidad con la anterior versión del paquete, definimos comandos similares a los de la primera versión; estos comandos tan solo se encargan de pasar su argumento a los comandos correspondientes que han sido creados en la nueva versión.

```
\newcommand\SetBgContents[1]{%
```

```
  \def\BackgroundContents{#1}}
\newcommand\SetBgAngle[1]{%
  \def\BackgroundAngle{#1}}
\newcommand\SetBgColor[1]{%
  \def\BackgroundColor{#1}}
\newcommand\SetBgScale[1]{%
  \def\BackgroundScale{#1}}
\newcommand\SetBgVshift[1]{%
  \def\BackgroundVShift{#1}}
\newcommand\SetBgHshift[1]{%
  \def\BackgroundHShift{#1}}
\newcommand\SetBgPosition[1]{%
  \def\BackgroundPosition{#1}}
\newcommand\SetBgAnchor[1]{%
  \def\BackgroundAnchor{#1}}
\newcommand\SetBgOpacity[1]{%
  \def\BackgroundOpacity{#1}}
```

Creamos ahora una copia de `\bg@material` para que pueda ser usada como comando de usuario.

```
\let\BgMaterial\bg@material
```

Finalmente, invocamos `\clearpage` al final del documento para prevenir documentos PDF corruptos cuando se compilan documentos de una sola página usando `\pdflatex`.

```
\AtEndDocument{\clearpage}
\endinput
```

2.4.3. Ejemplos de uso de la nueva versión del paquete

Para terminar este capítulo dedicado al paquete background, en esta sección presentamos los ejemplos de la sección 2.2.1, pero haciendo ahora uso de la nueva interfaz del paquete. Nuevamente, lipsum se utiliza exclusivamente para generar texto de manera automática.

Ejemplo 2.4.1

En el siguiente ejemplo se ilustra cómo usar el paquete background para producir un esquema personalizado para la numeración de páginas de un documento; comparar con el ejemplo 2.2.1. En la figura 2.1 (pág. 37) se muestra el resultado obtenido:

```
\documentclass{article}
\usepackage[placement=top,angle=0,%
  color=black!80,opacity=1,scale=4,hshift=60,vshift=-5]{background} \\
\backgroundsetup{contents={-\thepage-}}
\usepackage{lipsum}
```

Cuadro 2.3. Las opciones disponibles en la nueva versión del paquete `background`, sus posibles valores y los valores por defecto

Clave	Valores posibles	Valores por defecto
`pages`	`<all|||some>`	`all`
`placement`	`<center|||top|||bottom>`	`center`
`contents`	Texto, imágenes, etc.	`Draft`
`color`	Cualquier color válido	`red!45`
`angle`	Cualquier valor entre -360 and 360	60, para `center` 0, para `top` y `bottom`
`opacity`	Cualquier valor entre 0 y 1	0.5
`scale`	Cualquier valor positivo	15 para `center` 8 para `top` y `bottom`

`position`	Cualquier valos para la posición de un node	`current page.center` para `center`
		`current page.north` para `top`
		`current page.south` para `bottom`

`anchor`	Cualquier valos para el anclaje de un node	sin valor, para `center`
		`below` para `top`
		`above` para `bottom`

`hshift`	Cualquier valor	0
`vshift`	Cualquier valor	0

```
6
7  \pagestyle{empty}
8
9  \begin{document}
10 \lipsum[1-30]
11 \end{document}
```

☑

Ejemplo 2.4.2

En este ejemplo se muestra cómo proceder para obtener diferente material de fondo en las páginas pares e impares e un documento; comparar con el ejemplo 2.2.2. En la figura 2.2 (pág. 37) se muestra el resultado obtenido.

```
1  \documentclass{article}
2  \usepackage[contents={},opacity=1,scale=1.5,color=blue!90]{background}
3  \usepackage{lipsum}
4  \usepackage{ifthen}
5
6  \AddEverypageHook{%
7    \ifthenelse{\isodd{\thepage}}%
8      {\backgroundsetup{angle=90,position={0,-0.65\textheight},%
```

```
9         contents={The background package}}}%
10      {\backgroundsetup{angle=270,position={0.9\textwidth,-.7\textheight
          },%
11        contents={Version 2.0}}}%
12    \BgMaterial}
13
14  \begin{document}
15  \lipsum[1-30]
16  \end{document}
```

☑

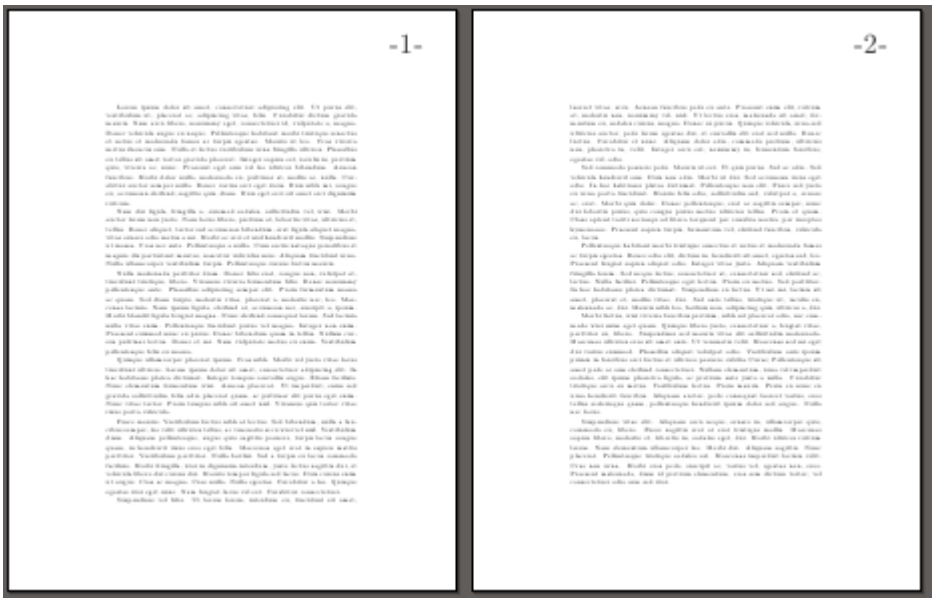

Figura 2.1. Esquema personalizado de numeración con el paquete `background`

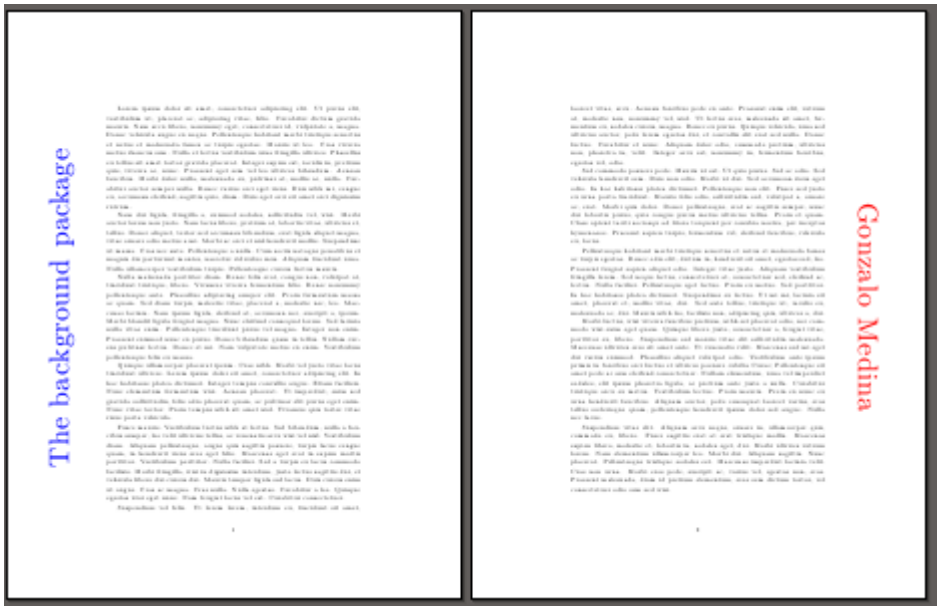

Figura 2.2. Diferente material de fondo para las páginas pares e impares utilizando el paquete `background`

3. El paquete fancypar

En este capítulo nos ocupamos del paquete LaTeX 2$_\varepsilon$ fancypar. En la primera sección describimos brevemente el paquete y, de manera suscinta, mencionamos otros paquetes actualmente disponibles en CTAN y que brindan una funcionalidad similar. En la sección 3.2 presentamos el código comentado de la versión actual del paquete y damos algunos ejemplos de su uso. En la sección 3.3 presentamos los cambios que han sido sugeridos para mejorar la implementación del paquete. Finalmente, en la sección 3.4 discutimos las estrategias usadas para implementar los cambios sugeridos, presentamos la versión comentada del código de la nueva versión del paquete e ilustramos, mediante ejemplos, el uso de la nueva versión mejorada del paquete.

3.1. Descripción del paquete

Existen algunos paquetes en CTAN que tienen una funcionalidad similar a la de fancypar: el paquete boites [20], creado por Vincent Zoonekynd, define entornos que admiten saltos de página dentro de cajas con marco con bordes que admiten decoraciones; el paquete framed [8], creado por Donald Arseneau, implementa tres entornos (framed, que enmarca la región correspondiente; shaded que colorea la región afectada; y leftbar, que coloca una línea vertical a la izquierda de la región). Los entornos admiten saltos de página. Este paquete implementa también un comando para crear entornos personalizados. Finalmente, está el paquete mdframed [17], creado por Elke Schubert y Marco Daniel, que ofrece una funcionalidad similar a la de framed, pero permitiendo al usuario controlar fácilmente los atributos de la región que será enmarcada o sombreada.

El paquete fancypar permite obtener párrafos de texto con decoraciones; el paquete incluye cinco estilos decorativos predefinidos e implementa además una serie de comandos simples que permiten al usuario definir sus propios estilos. Los cinco estilos predefinidos se controlan mediante sendos macros y pueden aplicarse a párrafos individuales de texto; los estilos predefinidos son: «cuaderno de texto», «cebra», «párrafo marcado», «párrafo con trazos discontinuos» y «párrafo subrayado». Ejemplos de los estilos y sus variantes pueden verse en las secciones 3.2.1 y 3.4.3.

La versión actualmente disponible del paquete posibilita controlar los atributos de cada uno de los estilos de una manera sencilla; cada estilo puede modificarse en cualquier parte del documento, haciendo uso de un sencillo mecanismo de tipo clave=valor implementado con el paquete xkeyval [7]. La implementación de fancypar hace uso de primitivas TeX (véanse [3] y [2]) para

manipular las líneas del párrafo.

3.1.1. Información general

- El paquete se encuentra en CTAN, bajo la licencia LPPL, desde abril de 2010: `http://www.ctan.org/pkg/fancypar`. En este momento se encuentra disponible la versión 1.1.
- El paquete se distribuye actualmente con MiKTeX y TeX Live (incluyendo, en particular, las distribuciones proTeXt y MacTeX).
- El paquete aparece reseñado en la revista TUGBOAT [1].
- El paquete se encuentra registrado bajo el nombre «The fancypar Package» ante la Oficina de Registro de la Dirección Nacional de Derechos de Autor del Ministerio del Interior y de Justicia, bajo el radicado 13-29-74 del 16 de mayo de 2011. Los derechos patrimoniales del paquete fueron cedidos a la Universidad Nacional de Colombia.
- El paquete incluye los archivos de código `fancypar.dtx` y `fancypar.ins`, el archivo de texto `README.txt`, el archivo generado `fancypar.sty` y el documento `fancypar.pdf` (la documentación del paquete).

3.2. Implementación actual del paquete

En esta sección se presenta el código comentado de la actual versión del paquete. Inicialmente, se hace la introducción estándar, anunciando que se trata de un paquete LaTeX 2_ε y se cargan los paquetes requeridos. La carga de estos paquetes se realiza solamente si no han sido cargados previamente por el usuario antes de requerir `fancypar`.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fancypar}[2010/04/25 v1.1 fancy paragraphs]
\end{lstliting}

\begin{lstlisting}
\@ifpackageloaded{xkeyval}{}{\RequirePackage{xkeyval}}
\@ifpackageloaded{tikz}{\usetikzlibrary{calc}}
  {\RequirePackage{tikz}\usetikzlibrary{calc}}
\@ifpackageloaded{xcolor}{}{\RequirePackage{xcolor}}
```

Declaramos ahora las opciones, con la ayuda del paquete `xkeyval`. Usamos `FP` como prefijo y `fancypar` como identificador de la familia. Las opciones `colorone`, `colortwo`, `textcolorone`, `textcolortwo`, `linecolor`, `intercolor`, `anchor`, `interheight`, `spiralcolor`, `textcolor`, `nbtextwidth`, `mark`, `rulecolor`, `separation`, `dashcolor` y `dashsymbol` se encargan de

definir sendos macros. En el cuadro 3.1 se muestran las opciones predefinidas para cada uno de los cinco estilos, junto con sus valores por defecto.

```
\DeclareOptionX[FP]<fancypar>{colorone}{\def\FancyZColorOne{#1}}
\DeclareOptionX[FP]<fancypar>{colortwo}{\def\FancyZColorTwo{#1}}
\DeclareOptionX[FP]<fancypar>{textcolorone}{\def\FancyZTextColorOne{#1}}
\DeclareOptionX[FP]<fancypar>{textcolortwo}{\def\FancyZTextColorTwo{#1}}
\DeclareOptionX[FP]<fancypar>{linecolor}{\def\FancyNlColor{#1}}
\DeclareOptionX[FP]<fancypar>{intercolor}{\def\FancyNilColor{#1}}
\DeclareOptionX[FP]<fancypar>{interheight}{\def\FancyNilHeight{#1}}
\DeclareOptionX[FP]<fancypar>{spiralcolor}{\def\FancyNSColor{#1}}
\DeclareOptionX[FP]<fancypar>{textcolor}{\def\FancyNTextColor{#1}}
\DeclareOptionX[FP]<fancypar>{nbtextwidth}{\def\FancyNTWidth{#1}}
\DeclareOptionX[FP]<fancypar>{mark}{\def\FancyMark{#1}}
\DeclareOptionX[FP]<fancypar>{rulecolor}{\def\FancyUColor{#1}}
\DeclareOptionX[FP]<fancypar>{separation}{\def\FancyDSeparation{#1}}
\DeclareOptionX[FP]<fancypar>{dashcolor}{\def\FancyDColor{#1}}
\DeclareOptionX[FP]<fancypar>{dashsymbol}{\def\FancyDSymbol{#1}}
```

Creamos dos longitudes auxiliares `\textindent` y `\textindentright` que controlarán la sangría dentro del estilo «cuaderno de texto». Inicialmente `\textindent` toma el valor `0pt` mientras que `\textindentright` toma el valor `\textwidth-2cm`.

```
\newlength\textindent{}
\newlength\textindentright{}
\setlength\textindentright{\textwidth}
\addtolength\textindentright{-2cm}
```

Definimos ahora la clave booleana `spiral` que se encargará de controlar si se dibuja o no la espiral en el estilo «cuaderno de texto». Dependiendo de si se ha de dibujar o no la espiral, se hacen los ajustes necesarios a `\textindent`.

```
\define@boolkey[FP]{fancypar}{spiral}{%
\ifFP@fancypar@spiral
  \setlength\textindent{-2.6mm}%
\else
  \setlength\textindent{3mm}%
\fi
}
```

Definimos a continuación la clave de escogencia `position`, con valores posibles `left` y `right`, que controlará la posición del símbolo que será usado como marca en el estilo «marcado».

```
\define@choicekey+[FP]{fancypar}{position}[\val\nr]{left,right}{%
  \ifcase\nr\relax
```

```
    \def\FancyMarkPosition{\llap{\mbox{\FancyMark\quad}}\box\linebox}
  \or
    \def\FancyMarkPosition{\box\linebox\rlap{\mbox{\quad\FancyMark}}}
  \fi
}{%
  \PackageWarning{fancypar}{erroneous input ignored}%
}
```

A continuación, utilizamos el macro `\DeclareOptionX*` con argumento `\CurrentOption` para producir un «warning» en el caso en que el usuario utilice una opción que no ha sido declarada.

```
\DeclareOptionX*{\PackageWarning{fancypar}{'\CurrentOption' ignored}}
```

Ahora creamos los colores que serán usados, por defecto, en los diferentes estilos. Los colores están dados siguiendo el esquema `RGB`. Aunque estos colores ya existen en el esquema `svgnames`, los definimos nuevamente para evitar tener que cargar el paquete `xcolor` con la opción `svgnames`, lo cual podría ocasionar conflictos si el usuario carga desprevenidamente el paquete con otras opciones.

```
\definecolor{zcolori}{RGB}{185,211,238}%SlateGray2
\definecolor{zcolorii}{RGB}{188,238,104}%DarkOliveGreen2
\definecolor{lcolor}{RGB}{159,182,205}%SlateGray3
\definecolor{scolor}{RGB}{205,205,180}%LightYellow3
\definecolor{rcolor}{RGB}{162,205,90}%DarkOliveGreen3
```

Ahora, usando `\ExecuteOptionsX`, asignamos valores por defecto a las claves que han sido credas usando `\DeclareOptionX` y luego procesamos los pares `<clave>=<valor>`.

```
\ExecuteOptionsX[FP]<fancypar>{%
  colorone=zcolori,%
  colortwo=zcolorii!90!white!70,%
  textcolorone=black,%
  textcolortwo=black,%
  linecolor=lcolor!80,%
  intercolor=green!20,%
  interheight=1pt,%
  spiralcolor=scolor,%
  spiral=true,%
  textcolor=black,%
  nbtextwidth=\textindentright,%
  mark=$\surd$,% \changes{v 1.1}{2010/04/25}{changed to $\surd$}
  rulecolor=rcolor,%
  position=right,%
  dashcolor=blue!50,%
```

```
  separation=0.9em,%
  dashsymbol=--
}

\ProcessOptionsX[FP]<fancypar>
```

Definimos ahora el comando de usuario `\fancyparsetup`. Este comando, se define como *robusto* y será el encargado de la modificación dinámica de los atributos de los estilos predefinidos. El comando recibe como argumento una lista de pares del estilo `<clave>=<valor>`, separados por comas, y modifica los atributos según los valores asignados, utilizado `\setkeys`. El cambio a los atributos surtirá efecto desde el punto en que se invocó el comando, hasta el final del documento, o hasta que se realice un nuevo llamado a `\fancyparsetup`.

```
\DeclareRobustCommand*\fancyparsetup[1]{%
  \setkeys[FP]{fancypar}{#1}
}
```

Definimos ahora sí los estilos predefinidos. Cada uno de los cinco macros asociados a los cinco estilos predefinidos recibe dos argumentos; en abstracto, la sintaxis para cada uno de los macros es:

```
\<Nombre>[<opciones>]{<texto>}
```

El primer argumento es opcional y se utiliza para modificar los atributos del estilo utilizando una lista de parejas `<clave>=<valor>`, separadas por comas; el segundo argumento es obligatorio y contiene el párrafo de texto al que se le aplicará el estilo. Los cinco macros utilizan los comandos `\FancyPreFormat` y `\FancyFormat` para aplicar línea a línea el estilo correspondiente.

Inicialmente definimos el macro `\NotebookPar`, para el estilo «cuaderno de notas». Este estilo pone el texto sobre un fondo que simula un cuaderno de notas, con perforaciones y espiral (esta espiral es opcional y puede desactivarse). El diseño de las perforaciones y de la espiral se hace con la ayuda del paquete PGF/TikZ:

```
\newcommand*\NotebookPar[2][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}
  \renewcommand\FancyPreFormat{\smallskip}
  \renewcommand\FancyFormat{%
    \hskip\textindent%
    \tikz{%
      \draw[draw=black,fill=white] (-1,-0.3) circle (3pt);%
      \ifFP@fancypar@spiral
        \draw[very thin,rotate=4,double=\FancyNSColor,%
          double distance=1.5pt]%
          (-1,-0.2) arc (40:-250:10pt and 2pt);%
```

```
      \else\relax
      \fi
    }
    \hskip4mm\vphantom{\strut}%
    \textcolor{\FancyNTextColor}{\box\linebox}%
    \color{\FancyNilColor}\hrule height\FancyNilHeight%
    \smallskip%
  }
  \setlength\parindent{0pt}
  \par\vskip\baselineskip
  \noindent%
  \begin{tikzpicture}[inner sep=-1.1pt]%
    \setlength\fboxsep{0pt}%
    \node (a) {\colorbox{\FancyNlColor}{%
      \vbox{%
        \vskip-0.5mm\parshape 1 0cm \FancyNTWidth%
        #2\par\add@fancy@format%
      }%
    }%
  } {};
  \end{tikzpicture}
  \par\bigskip
  \endgroup
}
```

Para el estilo «cebra» es necesario disponer de un contador para alternar los colores. Definimos entonces el contador `fancycount` dos comando de usuario `\FancyZColor` y `\FancyZTextColor` para controlar el color del fondo y el color del texto, respectivamente. Inicialmente estos comandos permiten alternar dos colores, pero pueden ser redefinidos para alternar cualquier cantidad de colores, lo cual permite obtener nuevos efectos interesantes. Ver el ejemplo 3.2.3 (pág. 50).

```
\newcounter{fancycount}
\newcommand*\FancyZColor{}
\renewcommand*\FancyZColor{%alternate line colors
  \ifodd\thefancycount %
    \FancyZColorOne%
  \else
    \FancyZColorTwo%
  \fi
}
\newcommand*\FancyZTextColor{}
\renewcommand*\FancyZTextColor{%alternate text colors
```

```
  \ifodd\thefancycount %
    \FancyZTextColorOne%
  \else
    \FancyZTextColorTwo%
  \fi
}
```

Definimos ahora al macro \ZebraPar, para el estilo «cebra». Este estilo pone el texto sobre un fondo de dos colores alternados, simulando un patrón al estilo cebra:

```
\newcommand*\ZebraPar[2][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}%
  \renewcommand\FancyPreFormat{\setcounter{fancycount}{0}}%
  \renewcommand\FancyFormat{%
    \noindent\stepcounter{fancycount}%
    \makebox[\textwidth]{\colorbox{\FancyZColor}{%
      \textcolor{\FancyZTextColor}{\box\linebox}}}%
    \hrule height 0pt%
  }
  \par\smallskip\noindent%
  \vbox{#2\par\add@fancy@format}%
  \par\smallskip%
  \endgroup
}
```

Para el estilo «párrafo con trazos discontinuos» usamos la primitiva \xleaders para repetir el patrón que producirá el trazo discontinuo utilizado. Redefiniendo el comando \FancyDSymbol se puede cambiar a voluntad el símbo usado:

```
\def\leaderfill{%
  \color{\FancyDColor}%
  \xleaders\hbox to \FancyDSeparation{\hss\FancyDSymbol\hss}\hfill%
}
```

Definimos el macro \DashedPar que se encarga de producir el estilo «párrafo con trazos discontinuos» ; al usar este estilo cada línea del párrafo aparece enmarcada en un patrón de trazos discontinuos:

```
\newcommand*\DashedPar[2][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}%
  \renewcommand\FancyPreFormat{%
    \hbox to \textwidth{\leaderfill}%
    \vskip-\baselineskip%
```

```
  }
  \renewcommand\FancyFormat{%
    \vphantom{\strut}\box\linebox%
    \hbox to \textwidth{\leaderfill}%
    \vskip-\baselineskip%
  }
  \par\medskip
  \vbox{\noindent#2\par\add@fancy@format\medskip}%
  \par\bigskip
  \endgroup
}%
```

Definimos el macro `\MarkedPar` que se encarga de producir el estilo «párrafo marcado» en el que cada línea es «marcada» mediante un símbolo cuya posición (constante para todo el párrafo) puede ser modificada por el usuario; el símbolo utilizado también puede cambiarse a voluntad:

```
\newcommand*\MarkedPar[2][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}
  \renewcommand\FancyPreFormat{}
  \renewcommand\FancyFormat{%
    \noindent%
    \FancyMarkPosition\par%
  }%
  \par\medskip%
  \vbox{#2\par\add@fancy@format}%
  \par\medskip%
  \endgroup
}
```

Por último, definimos el macro `\UnderlinededPar` que se encarga de producir el estilo «párrafo subrayado» en el que cada línea del párrafo recibe un subrayado continuo; el color de las líneas usadas para el subrayado puede ser modificado por el usuario:

```
\newcommand*\UnderlinedPar[2][]{
  \begingroup
  \setkeys[FP]{fancypar}{#1}
  \renewcommand\FancyPreFormat{}%
  \renewcommand\FancyFormat{%
    \box\linebox\color{\FancyUColor}\hrule
    \smallskip
  }
  \par\medskip%
  \vbox{\noindent#2\par\add@fancy@format}
```

```
  \par\medskip%
  \endgroup
}
```

Llegamos ahora al macro interno fundamental `\add@fancy@format` que hace uso de la técnica descrita en [2] para disecar párrafos. Se utilizan dos comandos de usuario: `\FancyPreFormat`, que permite controlar qué modificaciones se harán justo antes de aplicar el estilo y `\FancyFormat`, que se encarga de, línea a línea, aplicar el estilo deseado:

```
\newsavebox\linebox%
\def\add@fancy@format{%
  \setbox\linebox\lastbox
  \ifvoid\linebox\FancyPreFormat\else
    \unskip
    \unpenalty
    {\add@fancy@format}%
    \FancyFormat
  \fi
}
```

Para facilitar la creación de estilos personalizados por parte del usuario, definimos una copia del comando interno `\add@fancy@format`; esta copia servirá como comando de usuario:

```
\let\AddFancyFormat\add@fancy@format
```

Definimos finalmente los comandos de usuario que serán finalmente los responsables de producir el estilo: `\FancyPreFormat` y `\FancyFormat`. Inicialmente estos comandos no hacen nada; cada estilo se encarga de redefinirlos apropiadamente.

```
\newcommand\FancyPreFormat{}%
\newcommand\FancyFormat{}%
\endinput
```

3.2.1. Ejemplos de uso del paquete en su versión actual

En esta sección se presentan los estilos predefinidos por el paquete, así como los macros que permiten la fácil creación de nuevos estilos. Se ilustra el uso de los estilos y su personalización, así como del uso de los comandos mediante ejemplos sencillos. El paquete `lipsum` se utiliza solamente para generar texto de manera automática.

Los estilos predefinidos

Los cinco estilos predefinidos se activan mediante los macros
`\NotebookPar`, `\ZebraPar`, `\MarkedPar`, `\DashedPar` y `\UnderlinedPar`.

Cuadro 3.1. Resumen de los estilos predefinidos en la actual versión del paquete `fancypar`, junto con sus opciones y los valores por defecto

Comando/Estilo	Opciones	Valor por defecto
`\NotebookPar`	`linecolor=<color>`	`SlateGray3!80`
	`intercolor=<color>`	`green!20`
	`textcolor=<color>`	`black`
	`interheight=<length>`	`1pt`
	`spiralcolor=<color>`	`LightYellow3`
	`spiral=<true|false>`	`true`
	`nbtextwidth=<length>`	`\textindentright`
`\ZebraPar`	`colorone=<color>`	`SlateGray2`
	`colortwo=<color>`	`DarkOliveGreen2!90!white!70`
	`textcolorone=<color>`	`black`
	`textcolortwo=<color>`	`black`
`\DashedPar`	`separation=<length>`	`0.9em`
	`dashsymbol=<symbol>`	– (en-dash: `--`)
	`dashcolor=<color>`	`blue!50`
`\MarkedPar`	`mark=<symbol>`	`\surd` √
	`position=<right|left>`	`right`
`\UnderlinedPar`	`rulecolor=<color>`	`DarkOliveGreen3`

Ejemplo 3.2.1

Veamos los cinco estilos predefinidos en acción: la compilación del siguiente código permite ver el resultado de los valores por defecto para los estilos predefinidos:

```
\documentclass{article}
\usepackage{lipsum}

\begin{document}

\NotebookPar{\lipsum[1]}
\ZebraPar{\lipsum[1]}
\MarkedPar{\lipsum[1]}
\UnderlinedPar{\lipsum[1]}
\DashedPar{\lipsum[1]}

\end{document}
```

☑

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, √
adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consec- √
tetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique √
senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus √
sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida √
placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ul- √
trices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. √
Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan √
eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. √

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Variantes de los estilos predefinidos

Utilizando el argumento opcional declarado para cada uno de los macros que controla los estilos o el comando de usuario `\fancyparsetup` es posible personalizar los atributos de cada uno de ellos. A continuación se presentan algunos ejemplos, en todos ellos el preámbulo utilizado es

```
\documentclass{article}
\usepackage[x11names]{xcolor}
\usepackage{lipsum}
```

Ejemplo 3.2.2

Veamos una variante del estilo «cuaderno de notas», esta vez sin espiral y con cambio en los colores del fondo y del texto. La compilación del siguiente código

```
\NotebookPar[spiral=false,interheight=0pt,textcolor=Blue4,linecolor=
    yellow!30]{\lipsum[1]}
```

☑

produce el siguiente resultado:

○ Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut,
○ placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero,
○ nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.
○ Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis
○ egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum ur-
○ na fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien
○ est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices biben-
○ dum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla.
○ Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu,
○ accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Ejemplo 3.2.3

En este ejemplo se redefine el macro `\FancyZColor` para obtener un patrón tipo cebra, pero alternando tres colores para el fondo:

```
\renewcommand*{\FancyZColor}{
  \ifcase\intcalcMod{\value{fancycount}}{3}
    OliveDrab4!100!white!90\or Chocolate3!100!white!80
      \or LightGoldenrod3\fi
}

\ZebraPar{\lipsum[1]}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

☑

Ejemplo 3.2.4

Veamos ahora una variante al estilo «párrafo marcado». En el siguiente ejemplo utilizamos el comando `\fancyparsetup` para cambiar la marca utilizada y la posición en la que aparece:

```
1 \fancyparsetup{mark=$\bullet$,position=left}
2 \MarkedPar{\lipsum[1]}
```

- Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac,
- adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consec-
- tetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique
- senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus
- sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida
- placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ul-
- trices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla.
- Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan
- eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

☑

Ejemplo 3.2.5

Como último ejemplo, veamos cómo proceder a crear un nuevo estilo. El macro `\MiEstiloPar` que vamos a crear se encargará de que el texto de las líneas impares aparezca en color azul, mientras que el texto de las líneas pares tendrá color negro.

```
1 \documentclass{article}
2 \usepackage[english]{babel}
3 \usepackage{fancypar}
4
5 \newcounter{mycount}
6
```

```
\newcommand*\MiEstiloPar[1]{%
  \renewcommand\FancyPreFormat{\setcounter{mycount}{0}}
  \renewcommand\FancyFormat{%
    \stepcounter{mycount}
    \ifodd\themycount%
      \noindent\textcolor{blue}{\box\linebox}%
    \else%
      \box\linebox%
    \fi%
  }
  \par\medskip%
  \vbox{\noindent#1\par\AddFancyFormat}%
  \par\medskip%
}

\begin{document}
\MiEstiloPar{\lipsum[1]}
\end{document}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

☑

3.3. Cambios sugeridos

Durante el tiempo en que el paquete ha estado disponible como parte de MiKTeX y de TeX Live, el autor ha concebido una serie de cambios en la implementación del paquete tendientes a mejorar su desempeño, a brindar nuevas opciones y a hacerlo más amigable para el usuario; los cambios han sido incorporados en la nueva versión que se ha diseñado e implementado como parte de este trabajo son los siguientes:

1. La versión actual del paquete define los estilos decorativos mediante macros. Para dar mayor flexibilidad y comodidad sería conveniente poder aplicar los estilos no solo mediante macros

sino mediante entornos, permitiendo de esta manera la aplicación de un estilo a más de un párrafo simultáneamente así como el uso de material de tipo `verbatim`.

2. Actualmente los estilos solo se pueden aplicar a un párrafo de texto a la vez; ninguno de los macros admite el uso del comando `\par` dentro del argumento. Esta restricción debe eliminarse para ampliar el alcance y generalidad del paquete y evitar que el usuario deba realizar múltiples invocaciones de los comandos actuales.

3. Aunque el estilo «párrafo marcado» permite colocar la marca bien sea a la derecha o a la izquierda del texto que conforma el párrafo (en todo el documento) y cambiar este comportamiento individualmente, la versión actual del paquete no dispone de un mecanismo automatizado que permita poner la «marca» a la izquierda en las páginas pares y a la derecha en las páginas impares, lo cual es una alternativa tipográfica útil y necesaria en el caso de documentos tipo libro o de documentos com impresión a dos caras.

3.4. La nueva versión del paquete

En esta sección se discute la estrategia seguida para la implementación de los cambios sugeridos en la sección anterior; adicionalmente se presenta la versión comentada del código de la nueva versión del paquete. Finalmente se presentan algunos ejemplos que ilustran la interfaz de la nueva implementación y las mejoras realizadas al paquete `fancypar`.

3.4.1. Estrategia de implementación de los cambios

1. Para dar mayor flexibilidad y comodidad, en la nueva versión los estilos no solo están definidos mediante macros sino mediante entornos, permitiendo de esta manera la aplicación de un estilo a más de un párrafo simultáneamente así como el uso de material de tipo `verbatim`. Para evitar duplicidad en las definiciones, la estrategia utilizada es la implementación simultánea, para cada estilo, de un comando y un entorno, tal como se discutió en la sección 1.2 (pág. 13)

2. Una vez se dispone de entornos para los estilos, es posible usar el comando `\par` dentro de cada uno de los entorno. Sin embargo, esto hace necesaria una redefinición local del comando `\par`. Esta redefinición se hace en cada caso utilizando una copia de `\par` (ver la sección 1.4 (pág. 19)); esta copia se encarga de controlar la terminación de párrafos cuando se va a utilizar alguno de los entornos para aplicar el estilo a más de un párrafo.

3. Para permitir (en el estilo «párrafo marcado») la colocación automática de la marca a la izquierda en las páginas pares y a la derecha en las páginas impares, se hace necesario definir dos nuevas valores para la clave `position`. Dada la naturaleza asíncrona del mecanismo de

creación de páginas utilizado por TEX, es necesario además utilizar un mecanismo de tipo `\label`, `\ref` para evitar problemas con un posible posicionamiento incorrecto de la marca cuando un párrafo sea movido, por la rutina de salida de TEX, de una página a la siguiente. Para disponer de un mecanismo robusto, se utilizan los macros implementados en el módulo `zref-abspage` del paquete `zref` [14], creado por Heiko Oberdiek.

3.4.2. Código comentado de la nueva versión del paquete

Presentamos en esta sección el código comentado de la nueva versión del paquete. Los cambios introducidos se apoyan fuertemente en la utilización de algunos macros del «kernel» de LATEX y de otros implementados por el paquete `zref`.

Inicialmente, se hace la introducción estándar, anunciando que se trata de un paquete LATEX 2_ε y se cargan los paquetes requeridos, previa verificación de no haber sido cargados por el usuario antes de requerir `fancypar`.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fancypar}[2011/03/28 v2.0 fancy paragraphs]

\@ifpackageloaded{xkeyval}{}{\RequirePackage{xkeyval}}
\@ifpackageloaded{tikz}{\usetikzlibrary{calc}}
  {\RequirePackage{tikz}\usetikzlibrary{calc}}
\@ifpackageloaded{xcolor}{}{\RequirePackage{xcolor}}
\@ifpackageloaded{zref-abspage}{}{\RequirePackage{zref-abspage}}
```

La declaración de las opciones se sigue haciendo de la misma manera que en la versión actual.

```
\DeclareOptionX[FP]<fancypar>{colorone}{\def\FancyZColorOne{#1}}
\DeclareOptionX[FP]<fancypar>{colortwo}{\def\FancyZColorTwo{#1}}
\DeclareOptionX[FP]<fancypar>{textcolorone}{\def\FancyZTextColorOne{#1}}
\DeclareOptionX[FP]<fancypar>{textcolortwo}{\def\FancyZTextColorTwo{#1}}
\DeclareOptionX[FP]<fancypar>{linecolor}{\def\FancyNlColor{#1}}
\DeclareOptionX[FP]<fancypar>{intercolor}{\def\FancyNilColor{#1}}
\DeclareOptionX[FP]<fancypar>{interheight}{\def\FancyNilHeight{#1}}
\DeclareOptionX[FP]<fancypar>{spiralcolor}{\def\FancyNSColor{#1}}
\DeclareOptionX[FP]<fancypar>{textcolor}{\def\FancyNTextColor{#1}}
\DeclareOptionX[FP]<fancypar>{mark}{\def\FancyMark{#1}}
\DeclareOptionX[FP]<fancypar>{rulecolor}{\def\FancyUColor{#1}}
\DeclareOptionX[FP]<fancypar>{separation}{\def\FancyDSeparation{#1}}
\DeclareOptionX[FP]<fancypar>{dashcolor}{\def\FancyDColor{#1}}
\DeclareOptionX[FP]<fancypar>{dashsymbol}{\def\FancyDSymbol{#1}}
\DeclareOptionX[FP]<fancypar>{leftindent}{\def\FancyNleftindent{#1}}
\DeclareOptionX[FP]<fancypar>{rightindent}{\def\FancyNrightindent{#1}}
```

A diferencia de la versión actual, en la nueva versión la definición de la clave booleana `spiral` no se ocupa de hacer ajustes iniciales al sangrado. Estos ajustes se harán ahora en la definición del estilo «cuaderno de texto» directamente.

```
\define@boolkey[FP]{fancypar}{spiral}{}
```

Para la clave de escogencia `position` agregamos ahora dos nuevos valores: `inside` y `outside` que permitirán, en el estilo «párrafo marcado» colocar la marca en el margen externo (el izquierdo en las páginas pares y el derecho en las impares). El control sobre la paridad o imparidad de las páginas se realiza con la ayuda del contador `abspage` del paquete `zref`.

```
\define@choicekey+[FP]{fancypar}{position}[\val\nr]{left,right,inside,
    outside}{%
  \ifcase\nr\relax
    \def\FancyMarkPosition{\llap{\mbox{\FancyMark\quad}}\box\linebox}
  \or
    \def\FancyMarkPosition{\box\linebox\rlap{\mbox{\quad\FancyMark}}}
  \or
    \def\FancyMarkPosition{%
      \ifodd\value{abspage}
        \box\linebox\rlap{\mbox{\quad\FancyMark}}
      \else
        \llap{\mbox{\FancyMark\quad}}\box\linebox
      \fi
    }
  \or
    \def\FancyMarkPosition{%
      \ifodd\value{abspage}
        \llap{\mbox{\FancyMark\quad}}\box\linebox
      \else
        \box\linebox\rlap{\mbox{\quad\FancyMark}}
      \fi
    }
  \fi
}{%
  \PackageWarning{fancypar}{erroneous input ignored}%
}
```

A continuación, utilizamos el macro `\DeclareOptionX*` con argumento `\CurrentOption` para producir un «warning» en el caso en que el usuario utilice una opción que no ha sido declarada.

```
\DeclareOptionX*{\PackageWarning{fancypar}{'\CurrentOption' ignored}}
```

Creamos, como en la versión actual, los colores que serán usados, por defecto, en los diferentes estilos. Los colores están dados siguiendo el esquema `RGB`

```
\definecolor{zcolori}{RGB}{185,211,238}%SlateGray2
\definecolor{zcolorii}{RGB}{188,238,104}%DarkOliveGreen2
\definecolor{lcolor}{RGB}{159,182,205}%SlateGray3
\definecolor{scolor}{RGB}{205,205,180}%LightYellow3
\definecolor{rcolor}{RGB}{162,205,90}%DarkOliveGreen3
```

Ahora, usando `\ExecuteOptionsX`, asignamos valores por defecto a las claves que han sido credas usando `\DeclareOptionX` y luego procesamos los pares <clave>=<valor>.

```
\ExecuteOptionsX[FP]<fancypar>{%
  colorone=zcolori,%
  colortwo=zcolorii!90!white!70,%
  textcolorone=black,%
  textcolortwo=black,%
  linecolor=lcolor!80,%
  intercolor=green!20,%
  interheight=1pt,%
  spiralcolor=scolor,%
  spiral=true,%
  textcolor=black,%
  mark=$\surd$,% \changes{v 1.1}{2010/04/25}{changed to $\surd$}
  rulecolor=rcolor,%
  position=right,%
  dashcolor=blue!50,%
  separation=0.9em,%
  dashsymbol=--,
  leftindent=1cm,
  rightindent=1cm
}

\ProcessOptionsX[FP]<fancypar>
```

Ver el cuadro 3.2 en el que se muestran las opciones predefinidas en la nueva versión del paquete para cada uno de los cinco estilos, junto con sus valores por defecto.

Como en la versión anterior, definimos ahora el comando robusto `\fancyparsetup`. Este comando será el encargado de la modificación dinámica de los atributos de los estilos predefinidos. El comando recibe como argumento una lista de pares del estilo <clave>=<valor>, separada por comas, y modifica, por medio de `\setkeys`, los atributos según los valores asignados. El cambio a los atributos surtirá efecto desde el punto en que se invocó el comando, hasta el final del documento, o hasta que se realice un nuevo llamado a `\fancyparsetup`.

```
\DeclareRobustCommand*\fancyparsetup[1]{%
  \setkeys[FP]{fancypar}{#1}
```

```
}
```

Para que cada uno de los estilos admita el uso del comando `\par`, posibilitando así la aplicación de un estilo a más de un párrafo de manera simultánea, es necesario que cada estilo redefina localmente `\par`. Para este fin, creamos una copia de `\par` y la llamamos `\PAR`.

```
\let\PAR\par
```

A diferencia de la versión anterior, definimos ahora el comando `\FPNSpiral` que se encargará del trazado de los orificios y de la espiral, siempre y cuando la opción `spiral=true` está activa. El trazado hace uso de los constructos `circle` y `arc` del paquete PGF/TikZ.

```
\newcommand\FPNspiral{%
  \begin{tikzpicture}
    \draw[draw=black,fill=white] (-1,-0.3) circle (3pt);%
    \ifFP@fancypar@spiral
      \draw[very thin,rotate=4,double=\FancyNSColor,%
      double distance=1.5pt] (-1,-0.2) arc (50:-250:10pt and 2pt);%
    \else\relax\fi
  \end{tikzpicture}
}
```

Pasamos ahora a la nueva implementación de los cinco estilos predeterminados. Cada uno de los estilos se implementa ahora tanto como comando como entorno, haciendo uso de la técnica descrita en la sección 1.2 (pág. 13). Cada estilo se ocupa además de redefinir apropiadamente el comando `\PAR`.

Inicialmente, las definiciones necesarias para el estilo «cuaderno de notas»; la nueva definición es mucho más estructurada que la que se tiene en la versión actual; hacemos un control mucho mejor sobre el sangrado del texto y la forma de los párrafos y controlamos mejor la separación entre líneas consecutivas haciendo uso del comando primitivo `\offinterlineskip`; el posicionamiento de la espiral también se controla de una manera más precisa:

```
\newlength\FPNlinewidth
\newenvironment{NotebookPar}[1][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}%
  \setlength\FPNlinewidth\linewidth
  \addtolength\FPNlinewidth{-5.9pt}
  \addtolength\FPNlinewidth{-\FancyNleftindent}
  \addtolength\FPNlinewidth{-\FancyNrightindent}
  \renewcommand\FancyPreFormat{}%
  \renewcommand\FancyFormat{%
    \noindent\wd\linebox=\FPNlinewidth%
    \stepcounter{fancycount}%
```

```
    \offinterlineskip%
    {\colorbox{\FancyNlColor}{%
        \hspace*{16pt}\llap{\raisebox{-2pt}{\FPNspiral}}\hspace*{-16pt}%
        \textcolor{\FancyNTextColor}%
        {\strut\hspace*{\FancyNleftindent}\box\linebox\hspace*{\
            FancyNrightindent}}}}\\%
    \hbox{\color{\FancyNilColor}\rule{\linewidth}{\FancyNilHeight}}
  }
  \PAR\medskip\vskip\parskip
  % Allow for multiple paragraphs:
  \def\par{\parshape 1 0cm \FPNlinewidth\PAR\AddFancyFormat\egroup
    \offinterlineskip\vbox\bgroup}
  % Check if used as command or environment
  \@ifnextchar\bgroup
    % Command:
    {\long\def\@tempa####1{%
        \vbox\bgroup####1\endNotebookPar}\@tempa%
    }
     % Environment
    {\vbox\bgroup}%
}
{%
  \parshape 1 0cm \FPNlinewidth\PAR\AddFancyFormat\egroup\endgroup
}%
```

Como en la actual versión, definimos el contador `fancycount` y los dos comandos de usuario `\FancyZColor` y `\FancyZTextColor` para alternar los colores del texto y del fondo en el estilo «cebra»:

```
\newcounter{fancycount}
\newcommand*\FancyZColor{}
\renewcommand*\FancyZColor{%alternate line colors
  \ifodd\thefancycount %
    \FancyZColorOne%
  \else
    \FancyZColorTwo%
  \fi
}

\newcommand*\FancyZTextColor{}
\renewcommand*\FancyZTextColor{%alternate text colors
  \ifodd\thefancycount %
    \FancyZTextColorOne%
```

```
  \else
    \FancyZTextColorTwo%
  \fi
}
```

Ahora, las definiciones necesarias para el estilo «cebra»; la nueva definición utiliza la primitiva `\offinterlineskip` para permitir un mejor manejo del espacio entre líneas de un párrafo. En esta nueva versión tenemos que hacer un manejo más sofisticado del contador `fancycount` para alternar correctamente los colores al aplicar el estilo a más de un párrafo de texto simultáneamente:

```
\newenvironment{ZebraPar}[1][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}%
  \renewcommand\FancyPreFormat{}%
  \renewcommand\FancyFormat{%
    \noindent\stepcounter{fancycount}%
    \offinterlineskip%
    \makebox[\textwidth]{\colorbox{\FancyZColor}{%
      \textcolor{\FancyZTextColor}{\strut\box\linebox}}}
  }
  \PAR\medskip\vskip\parskip
  % Allow for multiple paragraphs:
  \def\par{\PAR\AddFancyFormat\egroup\offinterlineskip%
    \vbox\bgroup}
  % Check if used as command or environment
  \@ifnextchar\bgroup
    % Command:
    {\long\def\@tempa####1{%
        \vbox\bgroup####1\endZebraPar}\@tempa%
    }
     % Environment
    {\setcounter{fancycount}{0}\vbox\bgroup}%
}
{
  \PAR\AddFancyFormat\egroup\medskip\endgroup
}%
```

Como en la actual versión, para el estilo «párrafo con trazos discontinuos», usamos la primitiva `\xleaders` para repetir el patrón que producirá el trazo discontinuo utilizado. El comando `\FancyDSymbol` permite cambiar a voluntad el símbo usado:

```
\def\leaderfill{%
  \color{\FancyDColor}%
```

```
  \xleaders\hbox to \FancyDSeparation{\hss\FancyDSymbol\hss}\hfill%
}
```

Pasamos ahora al estilo «párrafo con trazos discontinuos»; aquí, la técnica usada es similar a la de la versión actual, salvo por la definición simultánea del entorno y del comando correspondientes y la redefinición interna de \par para el manejo apropiado de más de un párrafo de texto:

```
\newenvironment{DashedPar}[1][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}%
  \renewcommand\FancyPreFormat{%
    \hbox to \linewidth{\leaderfill}%
    \vskip-1.3\baselineskip%
  }
  \renewcommand\FancyFormat{%
    \vphantom{\strut}\box\linebox\vskip-4pt%
    \hbox to \linewidth{\leaderfill}%
    \vskip-1.3\baselineskip%
  }
  \PAR
  % Allow for multiple paragraphs:
  \def\par{\PAR\AddFancyFormat\medskip\egroup
    \PAR\bigskip\vbox\bgroup}
  % Check if used as command or environment
  \@ifnextchar\bgroup
    % Command:
    {\long\def\@tempa####1{\vbox\bgroup####1\endDashedPar}\@tempa}
    % Environment
    {\vbox\bgroup}%
}
{
  \PAR\AddFancyFormat
  \medskip\egroup%
  \PAR\bigskip
  \endgroup
}%
```

Para el estilo «párrafo marcado», utilizamos la misma estrategia que en la actual versión, salvo por la definición simultánea del entorno y del comando correspondientes y la redefinición interna de \par para el manejo apropiado de más de un párrafo de texto:

```
\newenvironment{MarkedPar}[1][]{%
  \begingroup
  \setkeys[FP]{fancypar}{#1}
```

```
  \renewcommand\FancyPreFormat{}
  \renewcommand\FancyFormat{%
    \noindent%
    \FancyMarkPosition\PAR%
  }%
  \PAR
  % Allow for multiple paragraphs:
   \def\par{\PAR\AddFancyFormat\vskip0pt}
  % Check if used as command or environment
  \@ifnextchar\bgroup
    % Command:
    {\long\def\@tempa####1{\vbox\bgroup####1\endMarkedPar}\@tempa}
    % Environment
    {\vbox\bgroup}%
}
{
  \PAR\AddFancyFormat\egroup%
  \PAR\endgroup
}%
```

Tenemos ahora la nueva implementación del estilo «párrafo subrayado». Para obtener un resultado óptimo al aplicar el estilo a más de un párrafo, la redefinición local de \par involucra un salto vertical negativo controlado por la longitud \bigskipamount:

```
\newenvironment{UnderlinedPar}[1][]{
  \begingroup
  \setkeys[FP]{fancypar}{#1}
  \renewcommand\FancyPreFormat{}%
  \renewcommand\FancyFormat{%
    {\vphantom{\mbox{}}\box\linebox}\vskip.2\smallskipamount%
      {\color{\FancyUColor}\hrule}
  }
  \PAR\vskip-\bigskipamount
  \def\par{\PAR\AddFancyFormat\vskip-\bigskipamount}
  % Check if used as command or environment
  \@ifnextchar\bgroup
    % Command:
    {\long\def\@tempa####1{\vbox\bgroup####1\endUnderlinedPar}\@tempa}
    % Environment
    {\vbox\bgroup}%
}
{
  \PAR\AddFancyFormat\egroup%
```

```
  \endgroup
}%
```

Nuevamente, tenemos el macro fundamental que se encarga de disecar los párrafos y de aplicar línea a línea el estilo escogido:

```
\newsavebox\linebox%
\def\add@fancy@format{%
  \setbox\linebox\lastbox
  \ifvoid\linebox\FancyPreFormat\else
    \unskip
    \unpenalty
    {\add@fancy@format}%
    \FancyFormat
  \fi
}
```

Definimos una copia del macro fundamental, para que sirva de comando de usuario:

```
\let\AddFancyFormat\add@fancy@format
```

Finalmente, definimos los comandos `\FancyPreFormat` y `\FancyFormat`. Inicialmente estod macros no hacen nada; cada estilo es el encargado de redefinirlos apropiadamente:

```
\newcommand\FancyPreFormat{}%
\newcommand\FancyFormat{}%
\endinput
```

3.4.3. Ejemplos de uso de la nueva versión del paquete

En esta sección ilustramos con dos ejemplos el uso de la versión mejorada del paquete `fancypar`; en particular, mostramos cómo ahora es posible aplicar los estilos de manera simultánea a varios párrafos de texto que pueden incluso contener material de tipo `verbatim`.

Ejemplo 3.4.1

En este ejemplo ilustramos el uso de la nueva opción `inside` para el estilo «párrafo marcado». Como marcas usamos dos comandos del paquete `bbding` que producen manos apuntanto a izquierda y a derecha. El estilo es aplicado a dos párrafos con la ayuda del entorno `MarkedPar`; el resultado de la compilación del código puede verse en la figura 3.2 (pág. 66):

```
\documentclass{article}
\usepackage{bbding}
\usepackage[position=inside]{fancypar}
\usepackage{lipsum}

```

Cuadro 3.2. Resumen de los estilos predefinidos en la nueva versión del paquete `fancypar`, junto con sus opciones y los valores por defecto

Estilo	Opciones	Valor por defecto
NotebookPar	linecolor=<color>	SlateGray3!80
	intercolor=<color>	green!20
	textcolor=<color>	black
	interheight=<length>	1pt
	spiralcolor=<color>	LightYellow3
	spiral=<true|false>	true
	nbtextwidth=<length>	\textindentright
ZebraPar	colorone=<color>	SlateGray2
	colortwo=<color>	DarkOliveGreen2!90!white!70
	textcolorone=<color>	black
	textcolortwo=<color>	black
DashedPar	separation=<length>	0.9em
	dashsymbol=<symbol>	– (en-dash: --)
	dashcolor=<color>	blue!50
MarkedPar	mark=<symbol>	\surd √
	position=<right|left|outside|inside>	right
UnderlinedPar	rulecolor=<color>	DarkOliveGreen3

```
\begin{document}

\lipsum[1]
\begin{MarkedPar}[mark=\HandRight]
\lipsum[1-2]
\end{MarkedPar}
\lipsum[3-4]
\begin{MarkedPar}[mark=\HandLeft]
\lipsum[5-7]
\end{MarkedPar}
\lipsum[8]

\end{document}
```

☑

Ejemplo 3.4.2

En este ejemplo mostramos la aplicación simultánea del estilo «cebra» a tres párrafos de texto utilizando el entorno `ZebraPar`; uno de los párrafos contiene material de tipo `verbatim`:

```
\documentclass{article}
\usepackage{fancypar}
```

```
\usepackage{lipsum}

\begin{document}

\begin{ZebraPar}[colorone=red!30,colortwo=blue!20]
\lipsum*[1]

Este p\'arrafo contiene material de tipo \texttt{verbatim}: \verb!a=b+c+
    d!

\lipsum*[1]
\end{ZebraPar}

\end{document}
```

El resultado de la compilación del código anterior puede verse en la figura 3.1 (pág. 65). ☑

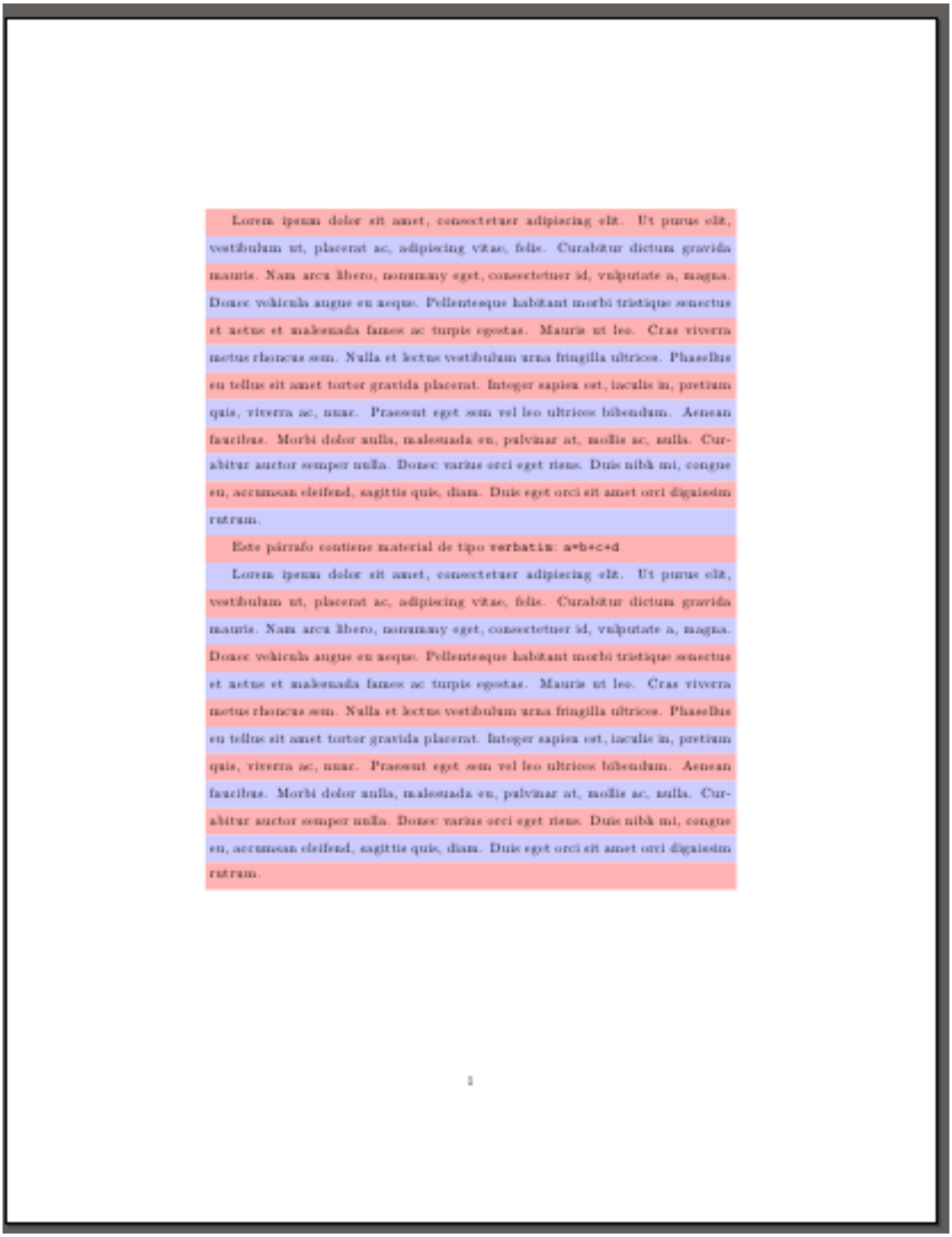

Figura 3.1. Ejemplo de aplicación simultánea a varios párrafos del nuevo entorno `ZebraPar`, para el estilo «cebra» del paquete `fancypar`. Los nuevos entornos admiten material de tipo `verbatim`

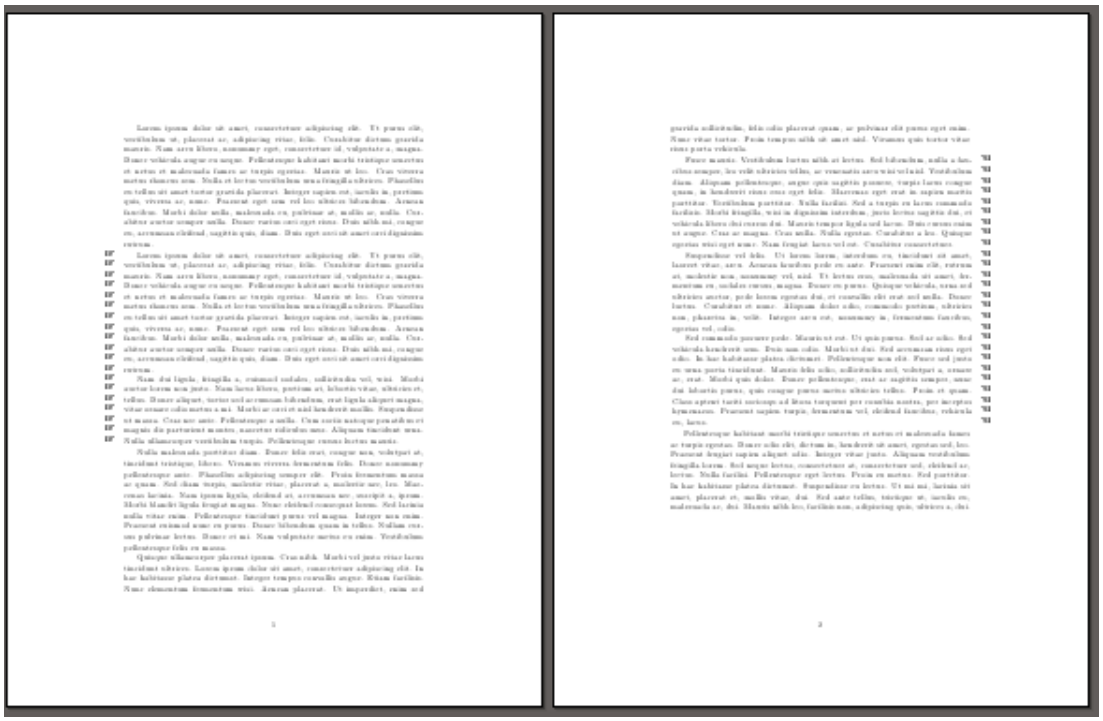

Figura 3.2. Ejemplo de aplicación simultánea a varios párrafos del nuevo entorno `MarkedPar` junto con una de sus nuevas opciones, para el estilo «párrafo marcado» del paquete `fancypar`

A. El código de las nuevas versiones de los paquetes background y fancypar

El código correspondiente a los archivos fuente `background.dtx` y de instalación `background.ins`, para la nueva versión del paquete `background`, y `fancypar.dtx` y de instalación `fancypar.ins`, para la nueva versión del paquete `fancypar` puede obtenerse en CTAN: `http://www.ctan.org/pkg/background` y `http://www.ctan.org/pkg/fancypar`, respectivamente.

Para generar los archivos subsidiarios `background.sty` y `fancypar.sty` debe abrirse una terminal y ejecutar

```
pdflatex background.ins
```

y

```
pdflatex fancypar.ins
```

Los archivos `.sty` generados deben copiarse en algún sitio en que TEX los encuentre (en el directorio apropiado del árbol TDS local, por ejemplo). Para generar la documentación (los archivos `background.pdf` y `fancypar.pdf`) debe ejecutarse

```
pdflatex background.dtx
```

y

```
pdflatex fancypar.dtx
```

Bibliografía

Libros y artículos

[1] Barbara Beeton, Karl Berry, editores. The treasure chest, en *TUGBOAT; The Communications of the TEX Users Group*. Volume 31, Number 1, 2010.
http://www.tug.org/TUGboat/Contents/contents31-1.html

[2] Victor Eijkhout. *TEX by Topic; A TEXnician's reference.* Addison-Wesley Publishing Company. Wokingham, England. 1992.

[3] Donald E. Knuth. *The TEXbook*. Addison-Wesley Profesional, 1984.

[4] Donald E. Knuth. *Literate programming*. The Computer Journal, 27(2):97–111, May 1984. British Computer Society.
Disponible en http://www.literateprogramming.com/knuthweb.pdf

[5] Frank Mittelblach, Michel Goossens, Jahannes Braams, David Carlisle, and Chris Rowley. *The LATEX Companion, Second Edition*. Addison Wesley, 2004.

Documentación en CTAN

[6] Equipo del Proyecto LATEX.
Documentación comentada del kernel de LATEX.
http://www.ctan.org/pkg/source2e

[7] Hendri Adriaens. El paquete xkeyval.
http://www.ctan.org/pkg/xkeyval

[8] Donald Arseneau. El paquete framed.
http://www.ctan.org/pkg/framed

[9] Sergio Callegari. El paquete draftwatermark.
http://www.ctan.org/pkg/draftwatermark

[10] Gonzalo Medina. El paquete background.
http://www.ctan.org/pkg/background

[11] Gonzalo Medina. El paquete fancypar.
http://www.ctan.org/pkg/fancypar

[12] Heiko Oberdiek. El paquete `letltxmacro`.
`http://www.ctan.org/pkg/letltxmacro`

[13] Heiko Oberdiek. El paquete `twoopt`.
`http://www.ctan.org/pkg/twoopt`

[14] Heiko Oberdiek. El paquete `zref`.
`http://www.ctan.org/pkg/zref`

[15] Scott Pakin. *How to Package Your LaTeX Package*. Noviembre, 2004.
Disponible en `http://www.tex.ac.uk/tex-archive/info/dtxtut/dtxtut.pdf`

[16] Alexander I. Rozhenko. El paquete `watermark`.
`http://www.ctan.org/pkg/watermark`

[17] Elke Schubert y Marco Daniel. El paquete `mdframed`.
`http://www.ctan.org/pkg/mdframed`

[18] Till Tantau. El paquete `pgf`.
`http://www.ctan.org/pkg/pgf`

[19] Jürgen Vollmer. El paquete `draftcopy`.
`http://www.ctan.org/pkg/draftcopy`

[20] Vincent Zoonekynd. El paquete `boites`.
`http://www.ctan.org/pkg/boites`