

Ejemplo básico

Comencemos con el clásico ejemplo:

```
<?php
require('fpdf.php');

$pdf=new FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(40,10,'¡Hola, Mundo!');
$pdf->Output();
?>
```

[\[Demo\]](#)

Después de incluir el fichero de la clase, creamos el objeto FPDF. El constructor [FPDF\(\)](#) se usa aquí con sus valores por defecto: las páginas son de tamaño a4 alargado y la unidad de medida es el milímetro. Se podría haber declarado explícitamente con:

```
$pdf=new FPDF('P','mm','A4');
```

Es posible usar el formato apaisado(L), otros formatos de página (como Carta y Legal) y otras unidades de medida (pt, cm, in).

Por el momento no hemos creado ninguna página, así que añadiremos una con [AddPage\(\)](#). El origen de coordenadas está en la esquina superior izquierda y la posición actual está por defecto situada a 1 cm de los bordes; los márgenes pueden cambiarse con [SetMargins\(\)](#).

Antes de que podamos imprimir texto, es obligatorio escoger una fuente con [SetFont\(\)](#), si no, el documento no será válido. Escogemos Arial en negrita de tamaño 16:

```
$pdf->SetFont('Arial','B',16);
```

Podríamos haber especificado itálica con I, subrayado con U o normal con una cadena vacía (o cualquier combinación de las anteriores). Observe que el tamaño de la fuente se determina en puntos, no en milímetros (ni en cualquier otra unidad establecida por el usuario); es la única excepción. Las otras fuentes incorporadas son Times, Courier, Symbol y ZapfDingbats.

Ahora podemos imprimir una celda con [Cell\(\)](#). Una celda es una superficie rectangular, con borde si se quiere, que contiene texto. Se imprime en la posición actual. Especificamos sus dimensiones, el texto (centrado o alineado), si queremos dibujar o no los bordes, y dónde se ubicará la posición actual después de imprimir la celda (a la derecha, debajo o al principio de la siguiente línea). Para añadir el borde, deberemos hacer esto:

```
$pdf->Cell(40,10,'¡Hola, Mundo!',1);
```

Para añadir una nueva celda a continuación, con el texto centrado e ir a la siguiente línea, escribiremos:

```
$pdf->Cell(60,10,'Hecho con FPDF.',0,1,'C');
```

Nota: el salto de línea puede provocarse también mediante [Ln\(\)](#). Este método le permite especificar además la altura del salto.

Finalmente, el documento se cierra y se envía al navegador con [Output\(\)](#). También podríamos haberlo guardado en un fichero pasando como parámetro el nombre del archivo.

Cuidado: en caso de que el PDF se envíe al navegador, nada más debe enviarse, ni antes ni después (el más mínimo espacio en blanco o retorno de carro también cuenta). Si se envía algún dato antes, obtendrá el mensaje de error: "Some data has already been output to browser, can't send PDF file". Si se envía después, su navegador puede que muestre únicamente una página en blanco.

Añadiendo nuevas fuentes y codificaciones

Este tutorial explica cómo usar fuentes TrueType o Type1 de forma que usted no se vea limitado a las fuentes incorporadas en FPDF. El otro punto de interés es que pueda elegir la codificación de la fuente, lo que le permitirá usar otros idiomas aparte de los europeos occidentales (ya que las fuentes estándar tienen muy pocos caracteres disponibles).

Existen dos formas de usar una nueva fuente: incluirla en el PDF o no. Cuando una fuente no se incluye, se busca en el sistema. La ventaja es que el fichero PDF es más pequeño; sin embargo, si la fuente no está disponible en el sistema, se usa otra como sustituta. Así que es aconsejable asegurarse de que la fuente en cuestión está instalada en los sistemas de los clientes. Si el fichero está destinado a una audiencia numerosas, es mejor incluir la fuente.

Añadir una nueva fuente requiere tres pasos para las TrueType:

- Generación del fichero de medidas (.afm)
- Generación del fichero de definición de la fuente (.php)
- Declaración de la fuente en el script

Para las fuentes Type1, el primer paso, teóricamente, no es necesario porque suelen venir acompañadas de un fichero AFM. En el caso de que usted sólo tenga un fichero de medidas en formato PFM, use el conversor disponible [aquí](#).

Generación del fichero de medidas

El primer paso para una TrueType consiste en generar el fichero AFM. Existe una aplicación para hacer esto: [ttf2pt1](#). El ejecutable para Windows está disponible [aquí](#). La instrucción para usarlo desde la línea de comandos es:

```
ttf2pt1 -a font.ttf font
```

Por ejemplo, para la Comic Sans MS Regular:

```
ttf2pt1 -a c:\windows\fonts\comic.ttf comic
```

Se crearán dos ficheros; el archivo en el que estamos interesados en el comic.afm.

Generación del fichero de definición de la fuente

El segundo paso consiste en generar un fichero PHP que contenga toda la información que necesita FPDF; además, el fichero de la fuente deberá comprimirse. Para hacer esto, se distribuye un script auxiliar en el directorio font/makefont del paquete: makefont.php. Contiene la siguiente función:

```
MakeFont(string fontfile, string afmfile [, string enc [, array patch  
[, string type]])
```

fontfile

Ubicación del fichero .ttf o .pfb.

afmfile

Ubicación del fichero .afm.

enc

Nombre de la codificación. Valor por defecto: cp1252.

patch

Modificación opcional de la codificación. Vacío por defecto.

type

Tipo de la fuente (TrueType o Type1). Valor por defecto: TrueType.

El primer parámetro es el nombre del fichero de la fuente. La extensión debe ser .ttf o .pfb y determina el tipo de fuente. Si tiene una fuente Type1 en formato ASCII (.pfa), puede convertirla a formato binario con [t1utils](#).

Si no quiere incluir la fuente en el documento, pase una cadena vacía. En este caso, el tipo se determina mediante el parámetro type.

Nota: en caso de que una fuente tenga el mismo nombre que una estándar, por ejemplo arial.ttf, es obligatorio incluirla. Si no lo hace, Acrobat usará su propia fuente (la estándar).

El fichero AFM es el que generamos anteriormente.

La codificación define la asociación entre un código (de 0 a 255) y un carácter. Los primeros 123 son fijos y se corresponden con los caracteres ASCII; los siguientes son variables. Las codificaciones se almacenan en ficheros .map. Están disponibles:

- cp1250 (Europa Central)
- cp1251 (cirílico)
- cp1252 (Europa Occidental)
- cp1253 (griego)
- cp1254 (turco)
- cp1255 (hebreo)
- cp1257 (báltico)
- cp1258 (vietnamita)
- cp874 (tailandés)
- ISO-8859-1 (Europa Occidental)
- ISO-8859-2 (Europa Central)
- ISO-8859-4 (báltico)
- ISO-8859-5 (cirílico)
- ISO-8859-7 (griego)
- ISO-8859-9 (turco)
- ISO-8859-11 (tailandés)
- ISO-8859-15 (Europa Occidental)
- ISO-8859-16 (Europa Central)
- KOI8-R (ruso)
- KOI8-U (ucraniano)

Por supuesto, la fuente debe contener los caracteres adecuados a la codificación escogida.

En el caso especial de una fuente de símbolos (esto es, que no contiene letras, como las fuentes Symbol o ZapfDingbats), pase una cadena vacía.

Las codificaciones que empiezan con cp son usadas por Windows; los sistemas Linux usan por lo general codificaciones ISO.

Nota: las fuentes estándar usan cp1252.

El cuarto parámetro le da la posibilidad de alterar la codificación. A veces puede que quiera añadir caracteres. Por ejemplo, ISO-8859-1 no contiene el símbolo para el euro. Para añadirlo en la posición 164, pase `array(164=>'Euro')`.

El último parámetro se usa para definir el tipo de la fuente en caso de que no se vaya a incluir (esto es, si el primer parámetro está vacío).

Después de llamar a la función (puede crear un nuevo fichero e incluir `makefont.php` o simplemente añadir la llamada en el propio `makefont.php`), se creará un fichero `.php` con el mismo nombre que el `.afm`. Puede renombrarlo si lo desea. En caso de incluir la fuente, el fichero se comprime y da lugar a un segundo fichero con extensión `.z` (excepto si la función de compresión no está disponible, puesto que requiere la biblioteca `zlib` para PHP). También puede renombrarlo, pero, en este caso, tendrá que modificar la variable `$file` en el fichero `.php` consecuentemente.

Ejemplo:

```
MakeFont('c:\\windows\\fonts\\comic.ttf','comic.afm','cp1252');
```

que devuelve los ficheros `comic.php` y `comic.z`.

Entonces tendrá que copiar los ficheros generados en el directorio de fuentes. Si el fichero de la fuente no pudo ser comprimido, copie el `.ttf` o `.pbf` en lugar del `.z`.

Declaración de la fuente en el script

El último paso es el más simple. Sólo necesita llamar al método [AddFont\(\)](#). Por ejemplo:

```
$pdf->AddFont('Comic','','comic.php');
```

o, simplemente,

```
$pdf->AddFont('Comic');
```

Y la fuente queda disponible (en los estilos normal y subrayado), lista para usar como las otras. Si hubiéramos trabajado con la Comic Sans MS Bold (`comicbd.ttf`), hubiésemos escrito:

```
$pdf->AddFont('Comic','B','comicbd.php');
```

Ejemplo

Veamos ahora un pequeño ejemplo completo. La fuente usada es Calligrapher, disponible en <http://www.abstractfonts.com/fonts/> (un sitio que ofrece un buen número de fuentes TrueType gratuitas). El primer paso es generar el AFM:

```
ttf2pt1 -a calligra.ttf calligra
```

que nos devuelve calligra.afm (y calligra.t1a, que podemos borrar). Generamos entonces la definición:

```
<?php
require('font/makefont/makefont.php');

MakeFont('calligra.ttf','calligra.afm');
?>
```

La función nos devolverá el siguiente informe:

Warning: character Euro is missing
Warning: character Zcaron is missing
Warning: character zcaron is missing
Warning: character eth is missing
Font file compressed (calligra.z)
Font definition file generated (calligra.php)

El carácter del euro no está en la fuente (es muy antigua). También faltan otros tres caracteres, pero no estamos interesados en ellos, así que da igual. Podemos copiar estos dos ficheros en el directorio de fuentes y escribir el script:

```
<?php
require('fpdf.php');

$pdf=new FPDF();
$pdf->AddFont('Calligrapher','','calligra.php');
$pdf->AddPage();
$pdf->SetFont('Calligrapher','',35);
$pdf->Cell(0,10,'Enjoy new fonts with FPDF!');
$pdf->Output();
?>
```

[\[Demo\]](#)

Acerca del símbolo del euro

El carácter del euro no aparece en todas las codificaciones, y no siempre está en la misma posición:

Codificación	Posición
cp1250	128
cp1251	136
cp1252	128
cp1253	128
cp1254	128
cp1255	128
cp1257	128
cp1258	128
cp874	128

ISO-8859-1	ausente
ISO-8859-2	ausente
ISO-8859-4	ausente
ISO-8859-5	ausente
ISO-8859-7	ausente
ISO-8859-9	ausente
ISO-8859-11	ausente
ISO-8859-15	164
ISO-8859-16	164
KOI8-R	ausente
KOI8-U	ausente

La codificación ISO-8859-1 está muy extendida, pero no incluye el símbolo del euro. Si lo necesita, la solución más simple consiste en usar cp1252 o ISO-8859-15, que son prácticamente iguales pero contienen el preciado símbolo.

Para la ISO-8859-2, es posible sustituirla por la ISO-8859-16, pero contiene muchas diferencias. Por tanto, es más sencillo apañar la codificación para añadir el símbolo, tal como se explica arriba. Lo mismo se aplica a las demás codificaciones.

Síntesis de fuentes bajo Windows

Cuando una fuente TrueType no está disponible en un estilo determinado, Windows es capaz de sintetizarla a partir de la versión normal. Por ejemplo, no hay Comic Sans MS en cursiva, pero puede ser construida a partir de la Comic Sans MS regular (normal). Esta característica puede ser empleada en un fichero PDF pero, desgraciadamente, requiere que la fuente normal esté instalada en el sistema (no debe incluirla en el documento). Así es como se hace:

- Generar el fichero de definición para la fuente normal sin incluirla en el documento (puede renombrarla para reflejar el estilo deseado)
- Abrirlo y añadir a la variable `$name` una coma (,) seguida del estilo deseado (*Italic*, **Bold** o **BoldItalic**)

Por ejemplo, para el fichero `comici.php`:

```
$name='ComicSansMS,Italic';
```

A partir de entonces, podrá usarse normalmente:

```
$pdf->AddFont('Comic','I','comici.php');
```

Reduciendo el tamaño de las fuentes TrueType

Los ficheros de fuentes son con frecuencia bastante voluminosos (más de 100, incluso 200 KB); esto se debe a que contienen los caracteres correspondientes a muchas codificaciones. La compresión `zlib` los reduce, pero continúan siendo bastante grandes. Existe una técnica para reducirlos aún más. Consiste en convertir la fuente a formato `Type1` con `ttf2pt1` especificando la codificación que le interesa; todos los demás

caracteres serán omitidos.

Por ejemplo, la fuente arial.ttf que viene con Windows 98 tiene un tamaño de 267 KB (contiene 1296 caracteres). Después de comprimirla, pesa 147 KB. Convirtámosla a Type1 manteniendo sólo los caracteres cp1250:

```
ttf2pt1 -b -L cp1250.map c:\windows\fonts\arial.ttf arial
```

Los ficheros .map están en el directorio font/makefont/ del paquete. El proceso devuelve arial.pfn y arial.afm. El fichero arial.pfb ocupa sólo 35 KB, 30 KB después de comprimirlo.

Es incluso posible ir más allá. Si sólo está usted interesado en un subconjunto de la codificación (es probable que no necesite los 217 caracteres), puede abrir el fichero .map y quitar las líneas que no le interesen. Consecuentemente, el tamaño del fichero disminuirá.

Columnas múltiples

En este ejemplo se mostrará como disponer texto en varias columnas.

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
    //Columna actual
    var $col=0;
    //Ordenada de comienzo de la columna
    var $y0;

    function Header()
    {
        //Cabecera
        global $title;

        $this->SetFont('Arial','B',15);
        $w=$this->GetStringWidth($title)+6;
        $this->SetX((210-$w)/2);
        $this->SetDrawColor(0,80,180);
        $this->SetFillColor(230,230,0);
        $this->SetTextColor(220,50,50);
        $this->SetLineWidth(1);
        $this->Cell($w,9,$title,1,1,'C',1);
        $this->Ln(10);
        //Guardar ordenada
        $this->y0=$this->GetY();
    }

    function Footer()
    {
        //Pie de página
        $this->SetY(-15);
        $this->SetFont('Arial','I',8);
        $this->SetTextColor(128);
        $this->Cell(0,10,'Página '.$this->PageNo(),0,0,'C');
    }

    function SetCol($col)
    {
        //Establecer la posición de una columna dada
        $this->col=$col;
        $x=10+$col*65;
        $this->SetLeftMargin($x);
        $this->SetX($x);
    }

    function AcceptPageBreak()
    {
        //Método que acepta o no el salto automático de página
        if($this->col<2)
        {
            //Ir a la siguiente columna
            $this->SetCol($this->col+1);
            //Establecer la ordenada al principio
        }
    }
}
```

```

        $this->SetY($this->y0);
        //Seguir en esta página
        return false;
    }
    else
    {
        //Volver a la primera columna
        $this->SetCol(0);
        //Salto de página
        return true;
    }
}

function ChapterTitle($num,$label)
{
    //Título
    $this->SetFont('Arial','',12);
    $this->SetFillColor(200,220,255);
    $this->Cell(0,6,"Capítulo $num : $label",0,1,'L',1);
    $this->Ln(4);
    //Guardar ordenada
    $this->y0=$this->GetY();
}

function ChapterBody($fichier)
{
    //Abrir fichero de texto
    $f=fopen($fichier,'r');
    $txt=fread($f,filesize($fichier));
    fclose($f);
    //Fuente
    $this->SetFont('Times','',12);
    //Imprimir texto en una columna de 6 cm de ancho
    $this->MultiCell(60,5,$txt);
    $this->Ln();
    //Cita en itálica
    $this->SetFont('','I');
    $this->Cell(0,5,'(fin del extracto)');
    //Volver a la primera columna
    $this->SetCol(0);
}

function PrintChapter($num,$title,$file)
{
    //Añadir capítulo
    $this->AddPage();
    $this->ChapterTitle($num,$title);
    $this->ChapterBody($file);
}

$pdf=new PDF();
$title='20000 Leguas de Viaje Submarino';
$pdf->SetTitle($title);
$pdf->SetAuthor('Julio Verne');
$pdf->PrintChapter(1,'UN RIZO DE HUIDA','20k_c1.txt');
$pdf->PrintChapter(2,'LOS PROS Y LOS CONTRAS','20k_c2.txt');
$pdf->Output();
?>

```

[Demo]

El método clave usado es [AcceptPageBreak\(\)](#). Permite aceptar o no el salto automático de línea. Evitándolo y alterando la posición actual y el margen, se consigue la disposición deseada en columnas.

Por lo demás, poco cambia; se han añadido dos propiedades (atributos) a la clase para almacenar el número de columna y la posición donde empiezan las columnas, y la llamada a MultCell() incluye un ancho de 6 centímetros.

Cabecera, pie, salto de página e imagen

Aquí tenemos un ejemplo de dos páginas con cabecera, pie de página y logotipo:

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
    //Cabecera de página
    function Header()
    {
        //Logo
        $this->Image('logo_pb.png',10,8,33);
        //Arial bold 15
        $this->SetFont('Arial','B',15);
        //Movernos a la derecha
        $this->Cell(80);
        //Título
        $this->Cell(30,10,'Title',1,0,'C');
        //Salto de línea
        $this->Ln(20);
    }

    //Pie de página
    function Footer()
    {
        //Posición: a 1,5 cm del final
        $this->SetY(-15);
        //Arial italic 8
        $this->SetFont('Arial','I',8);
        //Número de página
        $this->Cell(0,10,'Page '.$this->PageNo().'/{nb}',0,0,'C');
    }
}

//Creación del objeto de la clase heredada
$pdf=new PDF();
$pdf->AliasNbPages();
$pdf->AddPage();
$pdf->SetFont('Times','',12);
for($i=1;$i<=40;$i++)
    $pdf->Cell(0,10,'Imprimiendo línea número '.$i,0,1);
$pdf->Output();
?>
```

[Demo]

Este ejemplo hace uso de los métodos [Header\(\)](#) y [Footer\(\)](#) para procesar las cabeceras y pies de páginas. Se llaman automáticamente. Ya existen en la clase FPDF original, pero no hacen nada. Por ello, tenemos que heredar la clase y sobrescribirlos.

El logotipo se imprime con el método [Image\(\)](#) especificando su esquina superior izquierda y su anchura. La altura se calcula automáticamente respetando las proporciones de la imagen.

Para imprimir el número de página, se le pasa un valor nulo (null) como ancho de la celda. Eso significa que la celda se extenderá hasta el margen derecho de la página; puede ser útil centrar el texto. El número actual de la página se devuelve por el método [PageNo\(\)](#); mientras que el número total de páginas se obtiene mediante un valor especial de {nb} que será sustituido cuando se cierre el documento (suponiendo que usted antes utilizara [AliasNbPages\(\)](#)).

Observe el uso del método [SetY\(\)](#) que le permite especificar la posición en una ubicación absoluta respecto del origen de coordenadas de la página, empezando por el principio o por el final.

Otra característica interesante se usa en el ejemplo: el salto automático de página. Tan pronto como una celda cruza el límite máximo de la página (a 2 cm del final, por defecto), se ejecuta un salto y se recupera la fuente. Aunque la cabecera y el pie usan su propia fuente (Arial), el cuerpo del documento continua con Times. Este mecanismo automático de recuperación también se aplica a los colores y al ancho de línea. El límite que fuerza los saltos de página puede establecerse con [SetAutoPageBreak\(\)](#).

Enlaces y texto flotante

Este tutorial explica cómo incluir enlaces (internos y externos) y muestra una nueva manera de imprimir texto. También incluye un intérprete rudimentario de HTML.

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
    var $B;
    var $I;
    var $U;
    var $HREF;

    function PDF($orientation='P',$unit='mm',$format='A4')
    {
        //Llama al constructor de la clase padre
        $this->FPDF($orientation,$unit,$format);
        //Iniciación de variables
        $this->B=0;
        $this->I=0;
        $this->U=0;
        $this->HREF='';
    }

    function WriteHTML($html)
    {
        //Intérprete de HTML
        $html=str_replace("\n",' ', $html);
        $a=preg_split('/<(.*?)>/U', $html, -1, PREG_SPLIT_DELIM_CAPTURE);
        foreach($a as $i=>$e)
        {
            if($i%2==0)
            {
                //Text
                if($this->HREF)
                    $this->PutLink($this->HREF,$e);
                else
                    $this->Write(5,$e);
            }
            else
            {
                //Etiqueta
                if($e{0}=='/')
                    $this->CloseTag(strtoupper(substr($e,1)));
                else
                {
                    //Extraer atributos
                    $a2=explode(' ', $e);
                    $tag=strtoupper(array_shift($a2));
                    $attr=array();
                    foreach($a2 as $v)
                        if(ereg('^(\[|=]*)=("[\']*?([^"\']*?)("[\']*)?$',$v,$a3))
                            $attr[strtoupper($a3[1])]=$a3[2];
                    $this->OpenTag($tag,$attr);
                }
            }
        }
    }
}
```

```

    }
}

function OpenTag($tag,$attr)
{
    //Etiqueta de apertura
    if($tag=='B' or $tag=='I' or $tag=='U')
        $this->SetStyle($tag,true);
    if($tag=='A')
        $this->HREF=$attr['HREF'];
    if($tag=='BR')
        $this->Ln(5);
}

function CloseTag($tag)
{
    //Etiqueta de cierre
    if($tag=='B' or $tag=='I' or $tag=='U')
        $this->SetStyle($tag,false);
    if($tag=='A')
        $this->HREF='';
}

function SetStyle($tag,$enable)
{
    //Modificar estilo y escoger la fuente correspondiente
    $this->$tag+=( $enable ? 1 : -1);
    $style='';
    foreach(array('B','I','U') as $s)
        if($this->$s>0)
            $style.=$s;
    $this->SetFont('',$style);
}

function PutLink($URL,$txt)
{
    //Escribir un hiper-enlace
    $this->SetTextColor(0,0,255);
    $this->SetStyle('U',true);
    $this->Write(5,$txt,$URL);
    $this->SetStyle('U',false);
    $this->SetTextColor(0);
}

}

$html='Ahora puede imprimir fácilmente texto mezclando
diferentes estilos: <B>negrita</B>, <I>itálica</I>, <U>subrayado</U>,
o i
<B><I><U>todos a la vez</U></I></B>!<BR>
También puede incluir enlaces en el texto, como <A
HREF="http://www.fpdf.org">www.fpdf.org</A>,
o en una imagen: pulse en el logotipo.';

$pdf=new PDF();
//Primera página
$pdf->AddPage();
$pdf->SetFont('Arial','',20);
$pdf->Write(5,'Para saber qué hay de nuevo en este tutorial, pulse ');
$pdf->SetFont('','U');

```

```

$link=$pdf->AddLink();
$pdf->Write(5,'aquí',$link);
$pdf->SetFont('');
//Segunda página
$pdf->AddPage();
$pdf->SetLink($link);
$pdf->Image('logo.png',10,10,30,0,'','http://www.fpdf.org');
$pdf->SetLeftMargin(45);
$pdf->SetFontSize(14);
$pdf->WriteHTML($html);
$pdf->Output();
?>

```

[Demo]

El nuevo método para imprimir texto es [Write\(\)](#). Se parece mucho a [MultiCell\(\)](#); las diferencias son:

- El límite de la línea está en el margen derecho y la siguiente línea empieza en el izquierdo
- La posición actual se establece al final del texto

Así que le permite escribir un texto, alterar el estilo de la fuente, y continuar en el punto exacto donde lo dejó. Sin embargo, no puede justificar el texto simultáneamente a derecha y a izquierda.

Este método se usa en la primera página para añadir un enlace que apunta a la segunda página. El principio de la frase se escribe en un estilo normal, después cambiamos a subrayado y la terminamos. El enlace se crea con [AddLink\(\)](#), que devuelve el identificador del enlace. El identificador se pasa como tercer parámetro a [Write\(\)](#). Una vez que la segunda página se ha creado, usamos [SetLink\(\)](#) para hacer que el enlace apunte al principio de la página actual.

Después ponemos una imagen con un enlace en ella. Un enlace externo apunta a una URL (HTTP, mailto...). La URL se pasa como el último parámetro de [Image\(\)](#). Observe que los enlaces externos no funcionan cuando el PDF se muestra en un navegador Netscape.

Finalmente, el margen izquierdo se modifica después de la imagen con [SetLeftMargin\(\)](#) y se escribe texto en formato HTML. Se utiliza un intérprete HTML para esto, basado en la función de división mediante expresiones regulares [preg_split\(\)](#) y la opción [PREG_SPLIT_DELIM_CAPTURE](#) (incluida a partir de PHP 4.0.5) que permite recoger también los elementos de división (en este caso, las etiquetas). Si usted está usando una versión anterior de PHP, reemplace la línea con esta:

```

$a=preg_split('/[<>]/', $html);

```

que es menos restrictiva, pero devuelve el mismo resultado si se está tratando con HTML válido.

Las etiquetas reconocidas son [](#), [<I>](#), [<U>](#), [<A>](#) y [
](#); Las demás se ignoran. El intérprete también usa el método [Write\(\)](#). Se pone un enlace externo de la misma manera que uno interno (como tercer parámetro de [Write\(\)](#)).

Observe que [Cell\(\)](#) también permite incluir enlaces.

Tablas

Este tutorial se explicará como crear tablas fácilmente.

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
    //Cargar los datos
    function LoadData($file)
    {
        //Leer las líneas del fichero
        $lines=file($file);
        $data=array();
        foreach($lines as $line)
            $data[]=explode(';',chop($line));
        return $data;
    }

    //Tabla simple
    function BasicTable($header,$data)
    {
        //Cabecera
        foreach($header as $col)
            $this->Cell(40,7,$col,1);
        $this->Ln();
        //Datos
        foreach($data as $row)
        {
            foreach($row as $col)
                $this->Cell(40,6,$col,1);
            $this->Ln();
        }
    }

    //Una tabla más completa
    function ImprovedTable($header,$data)
    {
        //Anchuras de las columnas
        $w=array(40,35,40,45);
        //Cabeceras
        for($i=0;$i<count($header);$i++)
            $this->Cell($w[$i],7,$header[$i],1,0,'C');
        $this->Ln();
        //Datos
        foreach($data as $row)
        {
            $this->Cell($w[0],6,$row[0],'LR');
            $this->Cell($w[1],6,$row[1],'LR');
            $this->Cell($w[2],6,number_format($row[2]),'LR',0,'R');
            $this->Cell($w[3],6,number_format($row[3]),'LR',0,'R');
            $this->Ln();
        }
        //Línea de cierre
        $this->Cell(array_sum($w),0,'','T');
    }
}
```

```

//Tabla coloreada
function FancyTable($header,$data)
{
    //Colores, ancho de línea y fuente en negrita
    $this->SetFillColor(255,0,0);
    $this->SetTextColor(255);
    $this->SetDrawColor(128,0,0);
    $this->SetLineWidth(.3);
    $this->SetFont('', 'B');
    //Cabecera
    $w=array(40,35,40,45);
    for($i=0;$i<count($header);$i++)
        $this->Cell($w[$i],7,$header[$i],1,0, 'C',1);
    $this->Ln();
    //Restauración de colores y fuentes
    $this->SetFillColor(224,235,255);
    $this->SetTextColor(0);
    $this->SetFont('');
    //Datos
    $fill=0;
    foreach($data as $row)
    {
        $this->Cell($w[0],6,$row[0], 'LR',0, 'L', $fill);
        $this->Cell($w[1],6,$row[1], 'LR',0, 'L', $fill);
        $this->Cell($w[2],6,number_format($row[2]), 'LR',0, 'R', $fill);
        $this->Cell($w[3],6,number_format($row[3]), 'LR',0, 'R', $fill);
        $this->Ln();
        $fill=!$fill;
    }
    $this->Cell(array_sum($w),0, '', 'T');
}
}

$pdf=new PDF();
//Títulos de las columnas
$header=array('País', 'Capital', 'Superficie (km2)', 'Pobl. (en miles)');
//Carga de datos
$data=$pdf->LoadData('paises.txt');
$pdf->SetFont('Arial', '', 14);
$pdf->AddPage();
$pdf->BasicTable($header,$data);
$pdf->AddPage();
$pdf->ImprovedTable($header,$data);
$pdf->AddPage();
$pdf->FancyTable($header,$data);
$pdf->Output();
?>

```

[Demo]

Siendo una tabla un conjunto de celdas, lo natural es construirla de ellas. El primer ejemplo es el más básico posible: celdas con bordes simples, todas del mismo tamaño y alineadas a la izquierda. El resultado es algo rudimentario, pero es muy rápido de conseguir.

La segunda tabla tiene algunas mejoras: cada columna tiene su propio ancho, los títulos están centrados y el texto se alinea a la derecha. Más aún, las líneas horizontales se han eliminado. Esto se consigue mediante el parámetro `border` del método [Cell\(\)](#), que

especifica qué bordes de la celda deben imprimirse. En este caso, queremos que sean los de la izquierda (\mathbb{L}) y los de la derecha (\mathbb{R}). Seguimos teniendo el problema de la línea horizontal de fin de tabla. Hay dos posibilidades: o comprobar si estamos en la última línea en el bucle, en cuyo caso usaremos `LRB` para el parámetro `border`; o, como hemos hecho aquí, añadir la línea una vez que el bucle ha terminado.

La tercera tabla es similar a la segunda, salvo por el uso de colores. Simplemente hemos especificado los colores de relleno, texto y línea. El coloreado alternativo de las filas se consigue alternando celdas transparentes y coloreadas.

Saltos de línea y colores

Continuemos con un ejemplo que imprime párrafos justificados. También ilustra el uso de colores.

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
function Header()
{
    global $title;

    //Arial bold 15
    $this->SetFont('Arial','B',15);
    //Calculamos ancho y posición del título.
    $w=$this->GetStringWidth($title)+6;
    $this->SetX((210-$w)/2);
    //Colores de los bordes, fondo y texto
    $this->SetDrawColor(0,80,180);
    $this->SetFillColor(230,230,0);
    $this->SetTextColor(220,50,50);
    //Ancho del borde (1 mm)
    $this->SetLineWidth(1);
    //Título
    $this->Cell($w,9,$title,1,1,'C',1);
    //Salto de línea
    $this->Ln(10);
}

function Footer()
{
    //Posición a 1,5 cm del final
    $this->SetY(-15);
    //Arial itálica 8
    $this->SetFont('Arial','I',8);
    //Color del texto en gris
    $this->SetTextColor(128);
    //Número de página
    $this->Cell(0,10,'Página '.$this->PageNo(),0,0,'C');
}

function ChapterTitle($num,$label)
{
    //Arial 12
    $this->SetFont('Arial','',12);
    //Color de fondo
    $this->SetFillColor(200,220,255);
    //Título
    $this->Cell(0,6,"Capítulo $num : $label",0,1,'L',1);
    //Salto de línea
    $this->Ln(4);
}

function ChapterBody($file)
{

```

```

//Leemos el fichero
$f=fopen($file,'r');
$txt=fread($f,filesize($file));
fclose($f);
//Times 12
$this->SetFont('Times','',12);
//Imprimimos el texto justificado
$this->MultiCell(0,5,$txt);
//Salto de línea
$this->Ln();
//Cita en itálica
$this->SetFont('','I');
$this->Cell(0,5,'(fin del extracto)');
}

function PrintChapter($num,$title,$file)
{
    $this->AddPage();
    $this->ChapterTitle($num,$title);
    $this->ChapterBody($file);
}
}

$pdf=new PDF();
$title='20000 Leguas de Viaje Submarino';
$pdf->SetTitle($title);
$pdf->SetAuthor('Julio Verne');
$pdf->PrintChapter(1,'UN RIZO DE HUIDA','20k_c1.txt');
$pdf->PrintChapter(2,'LOS PROS Y LOS CONTRAS','20k_c2.txt');
$pdf->Output();
?>

```

[Demo]

El método [GetStringWidth\(\)](#) le permite determinar la longitud de una cadena en el tipo de letra actual, y se usa aquí para calcular la posición y ancho del borde que rodea al título. Después se establecen los colores (mediante [SetDrawColor\(\)](#), [SetFillColor\(\)](#) y [SetTextColor\(\)](#)) y el borde de la línea se establece en 1 mm (en contra de los 0,2 por defecto) con [SetLineWidth\(\)](#). Finalmente, imprimimos la celda (el último parámetro a 1 indica que debe colorearse el fondo).

El método usado para imprimir los párrafos es [MultiCell\(\)](#). Cada vez que la línea llega al extremo derecho de la celda o aparece un carácter de fin de línea, se ejecuta un salto de línea y se crea automáticamente otra celda debajo de la actual. El texto se encuentra justificado por defecto.

Se definen dos propiedades del documento: título ([SetTitle\(\)](#)) y autor ([SetAuthor\(\)](#)). Las propiedades pueden verse de dos maneras. La primera es abrir directamente el documento con Acrobat Reader, irse al menú Archivo, Información del documento, General. La segunda, también disponible por la extensión (plug-in) de Acrobat Reader, es pulsar en el triángulo situado inmediatamente encima de la barra de desplazamiento de la derecha y elegir Información del documento.