

Sobres impresos con Perl y PostScript

Cartas Postales Perfectas

No necesitamos ni un paquete Office ni LaTeX para imprimir los sobres de un mailing. Los módulos Perl de PostScript, una base de datos y el listado de este artículo forman una solución perfecta para crear sobres para envíos masivos de correo.

POR MICHAEL SCHILLI



Linux puede hacer más o menos cualquier cosa. Leer imágenes digitales desde cámaras, reproducir pistas digitales, escribir CDs, incluso conseguir que un escáner USB funcione (con algo de ayuda de *Xsane*). Pero a título personal tenía que utilizar Windows de vez en cuando para realizar una tarea en concreto: a la hora de remitir varias cartas iguales a distintos destinatarios, las direcciones de los cuales se extraía de una base de datos, tenía que salirme de mi adorado Linux, rearrancar el ordenador, no distraerme ni ir a por un café mientras la máquina arrancaba para

no perderme, el menú de arranque, y meterme en mi semi-olvidada instalación de Windows, desde el cual, por supuesto, no tenía acceso ni a la mitad de las herramientas a las que Linux me tiene acostumbrado y sin las cuales me siento totalmente desamparado. Un montón de trabajo para imprimir unos sobres. Solía usar un viejo programa de Windows para esta tarea, pero eso se acabó. Existe un utilísimo programa bajo Linux que llamado Ghostscript que puede convertir nuestra vieja impresora doméstica en una fabulosas máquina PostScript.

Si nuestra distribución no se ha ocupado ya de esto, véase [1] para saber como se hace. Como muestra la figura 1, la exportación de datos desde la base de

datos Windows no fue un problema a usar el formato de separadores mediante comas (CSV). Todo lo que tuve que hacer a continuación fue generar un archivo PostScript para cada sobre y luego enviar esos archivos a la impresora. Y esto es un juego de niños con módulos CPAN como *PostScript::File* y *PostScript::TextBlock*, como [2] nos dirá. PostScript es básicamente otro lenguaje de programación. Los archivos PostScript están hechos de texto ASCII legible y contienen una lista de comandos necesarios para generar una página impresa.

Pintando mediante números

No obstante, PostScript usa el llamado sistema de coordenadas matemáticas y

THE AUTHOR

Michael Schilli trabaja como ingeniero Web en la empresa AOL/Netscape en Mountain View, California. Escribió "Perl Power" for Addison-Wesley y puede ser contactado en la dirección mschilli@perlmeister.com. Su página web es <http://perlmeister.com>.

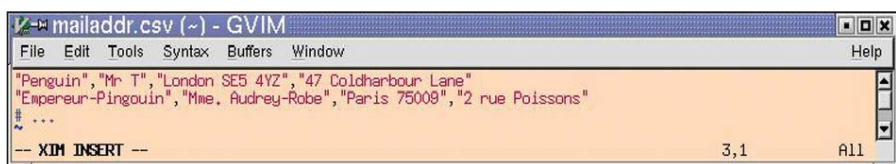


Figura 1: Los campos en el archivo de dirección separados por comas.

esto algo muy poco habitual en un programa de composición. El origen del sistema coordenadas es la esquina inferior izquierda de la página. Los valores positivos del eje *x* arrancan desde la izquierda y se extienden hacia la derecha y los valores positivos de *y* van incrementando de valor de abajo arriba, tal y como nos enseñó nuestro viejo maestro de matemáticas en nuestro colegio de secundaria. PostScript usa puntos pica o PostScript, que miden aproximadamente 1/3 de milímetro. Como ejemplo de lo que se puede conseguir con PostScript, los siguientes comandos imprimirán el nombre *John Doe* en el recuadro de la dirección de un sobre:

```
0 setgray 401.95 156 moveto
/Helvetica-iso findfont
18 scalefont setfont
(John Doe) show
```

Empezando en la esquina izquierda inferior del sobre, nos iríamos casi 402 puntos hacia la izquierda y 156 hacia arriba para imprimir las letras que ven entre los paréntesis en la fuente especificada (Helvetica-iso) y con el tamaño especificado (18 puntos) y de izquierda a derecha del papel.

Para empezar a programar con PostScript bajo Perl, CPAN proporciona los módulos *PostScript::File* y *PostScript::TextBlock* para simplificar el asunto. El primero sirve para insertar la cabecera PostScript de la siguiente manera:

```
%!PS-Adobe-3.0
```

lo cual se ocupa de cosas como la orientación de la página, los bordes y el orden de la página. *PostScript::TextBlock* acepta líneas múltiples y escribe desde la coordenadas indicadas. No obstante podemos esperar muchas horas de diversión realizando ajustes en los parámetros de cada módulo hasta que consigamos producir el formato deseado en el lugar adecuado de la página.

Normalmente se utiliza el siguiente formato: el bloque de texto con la dirección del remitente comienza aproximadamente 20 milímetros por encima de la esquina inferior derecha en las direcciones *x* e *y*. En otras palabras, nosotros no queremos especificar el punto de ini-

cio del campo dirección, si no que indicamos la posición de la esquina derecha inferior del bloque de texto. Esto nos asegura que el bloque siempre acabará en la misma posición al margen de las variables que usemos o la larga que sea la dirección.

En cuanto al programa, el código de ejemplo que se muestra en el listado 1 define una constante *Sender* (línea 14). Ésta lee los datos de un archivo *.csv* e imprime un sobre como los mostrados en las figuras 2 ó 3 para cada dirección que encuentre.

Importación desde Windows

\$ADDR_CSV en la línea 13 especificamos el nombre del archivo de direcciones que debe tener la misma estructura que el mostrado en la figura 2. El comando para enviar un archivo PostScript a nuestra impresora se define en la línea 17 (*\$PRINT_CMD*). Si deseamos hacer una prueba sin malgastar papel sustituiremos "*lpr*" por "*ghostview*" para enviar sobres virtuales para que se visualicen en nuestra pantalla.

El código de la línea 19 abre el archivo de direcciones y el bloque *while* que comienza en la línea 22 se repite con las entradas que son analizadas usando expresiones regulares. En lugar de esta técnica podríamos haber usado el módulo *CPAN Text::CSV_XS*, aunque esto podría haber sido excesivo para las direcciones de nuestro ejemplo, pues son extremadamente simples y no tienen complicaciones como comillas o comas embebidas.

Trabajo Manual vs Automatizado

La línea 23 interpreta cualquier línea que comienza con el símbolo *#* y un espacio en blanco como comentarios. Esto es muy útil si solo queremos imprimir algunas entradas. Facilita muchísimo la tarea poder "comentar" y excluir todas las entradas que queramos evitar imprimir con el símbolo *#*. El comando *split* de la línea 24 divide líneas donde aparecen comas separadoras. *map* elimina las comillas dobles. Como la sustitución *s//g;* no devuelve una cadena como resultado válido, *\$_*; simplemente se añade.

La línea 27 crea un objeto *PostScript::File* que usa la palabra clave *landscape* para rotar el formato de la página. *reencode => 'ISOLatin1Encoding'* proporciona soporte para todos los caracteres Latin1. El formato del sobre se fija en *Envelope-DL*. Si necesitamos un formato diferente para utilizar un sobre de distinto tamaño no tendremos ninguna dificultad para modificar el archivo. Un sobre DIN A6 tienen unas medidas aproximadas de 10.47 centímetros de alto por 14.81 centímetros de ancho. Para que esto se aplique en nuestro programa, utilizaremos el siguiente bloque de código:

```
my $ps = new PostScript::File(
    landscape => 1,
    reencode => 'ISOLatin1Encoding',
    width => cm(10.47),
    height => cm(14.81),
);
```



Figura 2: No importa si el remitente tiene un nombre corto...

LISTADO 1: SOBRE

```

001 #!/usr/bin/perl
002 #####
003 # sobre - Papel de impresión
sobres
004 # Mike Schilli, 2003
(m@perlmeister.com)
005 #####
006 use warnings;
007 use strict;
008
009 use PostScript::File;
010 use PostScript::TextBlock;
011 use File::Temp qw(tempfile);
012
013 my $ADDR_CSV =
"mailaddr.csv";
014 my $SENDER = q{Steven
Sender,
015 9 Sender Street,
016 San Francisco, CA 94107};
017 my $PRINT_CMD = "lpr";
018
019 open FILE, $ADDR_CSV or
020 die "Cannot open $ADDR_CSV";
021
022 while(<FILE>) {
023 next if /^\\s*#/;
024 my @addr = split /,/;
025 @addr = map { s/"//g; }
@addr;
026
027 my $ps = PostScript::File-
>new(
028 landscape => 1,
029 reencode
=>'ISOLatin1Encoding',
030 paper => "Envelope-DL",
031 );
032
033 my ($tmp_fh, $tmp_file) =
034 tempfile(SUFFIX => ".ps");
035
036 my($last, $first, $city,
$str) = @addr;
037
038 # Remitente
039 my($bw, $bh, $b) =
textbox($SENDER,
040 "Helveticaiso", 10, 12);
041 my ($code) = $b->Write($bw,
$bh, cm(2),
042 $ps->get_width() - cm(2));
043 $ps->add_to_page($code);
044
045 # Destinatario
046 my $to = "$first
$last\n$str\n\n$city\n";
047 ($bw, $bh, $b) =
textbox($to,
048 "Helveticaiso", 18, 20);
049 ($code) = $b->Write($bw,
$bh,
050 $ps->get_height()- $bw -
cm(2),
051 $bh + cm(2));
052 $ps->add_to_page($code);
053
054 # Imprimir a archivo
temporal
055 (my $base = $tmp_file) =~s/
\\.ps$/;
056 $ps->output($base);
057
058 # Enviar a impresora
059 system("$PRINT_CMD
$tmp_file") and
060 die "$PRINT_CMD $tmp_file:
$!";
061
062 # Borrar
063 unlink "$tmp_file" or
064 die "Cannot unlink
$tmp_file: $!";
065 }
066 #####
067 sub textbox {
068 #####
069 #####
070 my($text, $font, $size,
$leading) = @_;
071
072 my $b = PostScript::
TextBlock->new();
073
074 $b->addText(
075 font => $font,
076 text => $text,
077 size => $size,
078 leading => $leading);
079
080 return(tb_width($text,
$font, $size),
081 tb_height($text, $leading),
082 $b);
083 }
084
085 #####
086 sub cm {
087 #####
088 return int($_[0]*72/2.54);
089 }
090
091 #####
092 sub tb_width {
093 #####
094 my($text, $font, $size)= @_;
095
096 $font =~ s/-iso//;
097
098 my $max_width = 0;
099
100 for(split /\n/, $text) {
101 s/[äöüÜß]/A/ig;
102 my $w =
103 PostScript::Metrics::
stringwidth(
104 $_, $font, $size);
105 $max_width = $w if $w >
$max_width;
106 }
107
108 return $max_width;
109 }
110
111 #####
112 sub tb_height {
113 #####
114 my($text, $leading) = @_;
115
116 my $lines = 1;
117 $lines++ for $text =~/\n/g;
118
119 return $lines*$leading;
120 }

```

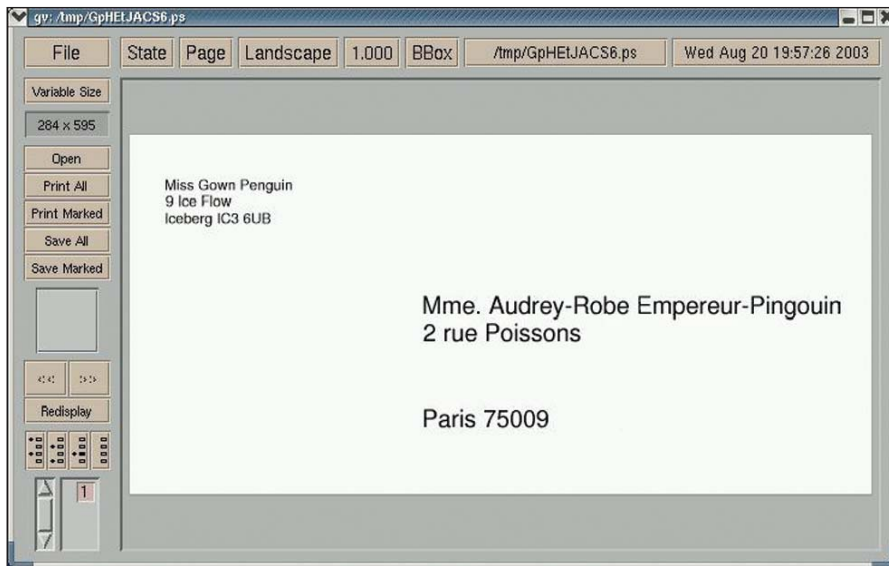


Figura 3: ... o largo, el offset se mantiene inalterado

La línea 36 guarda los campos de dirección en las variables *\$last*, *\$first*, *\$city* y *\$str*. La línea 39 llama a la función *textbox()* (ver una descripción en detalles de esta función más adelante), que espera unas líneas que contengan el nombre de la fuente y su tamaño y el espaciado interlineal en puntos PostScript. Hemos decidido usar *Helvetica-iso* por que está instalada por defecto. El sufijo *-iso* también soporta caracteres acentuados. *textbox()* nos devuelve 3 valores: un objeto *PostScript::TextBlock* y el ancho y alto de la caja de texto generada en puntos PostScript.

Más adelante, la línea 41 llama al método *Write()* usado por el objeto *PostScript::TextBlock* para crear el código PostScript. *Write()* espera 4 parámetros: ancho y alto del bloque de texto y las distancias en *x* e *y* desde el origen. La anchura y altura las proporciona la función *textbox()* llamada con anterioridad.

Pensando en Offsets

El offset en *x* (la distancia desde el margen izquierdo) es de unos 2 centímetros, que puede ser expresada utilizando la función *cm()*, definida más abajo para convertir centímetros a puntos PostScript. El offset en *y* es más complicado puesto que *Write()* espera una distancia desde el margen inferior cuando nosotros necesitamos 2 centímetros desde el margen superior. No debemos preocuparnos puesto que el método *\$ps->get_width()* del objeto *PostScript::File* nos proporciona la altura del

sobre y simplemente debemos restar *cm(2)* de esta en la línea 42.

Debemos saber que *PostScript::File* mantiene la noción de ancho y alto al margen de que usemos el modo *landscape* en el que el papel se rota 90°. En nuestro caso, *get_width()* nos muestra la altura y *get_height()* la anchura. *Write* devuelve una lista en la que el primer elemento es el código PostScript del bloque de texto. La línea 43 añade este código a la página PostScript actual.

Se utiliza el mismo procedimiento para el destinatario: la línea 46 concatena nombre y apellidos, la calle y la ciudad. La función *adresseetextbox()* usa una fuente un poco mayor al igual que con el espacio interlineal. El offset en *x* desde la esquina superior izquierda de la caja de texto hasta el origen del PostScript se calcula restando a la longitud del sobre (*\$ps->get_height()*) el ancho de la caja de texto (*\$bw*) menos 2 centímetros de borde (*cm(2)*). El offset en *y* es la distancia desde el borde superior de la caja de texto hasta el límite inferior del sobre y es el resultado de sumar 2 centímetros a la altura de la caja de texto (*\$bh + cm(2)*).

Vida corta

La función *tempfile()* del módulo *File::Temp* crea un archivo temporal con el sufijo PostScript *.ps* en la línea 34 y devuelve un handle de archivo que admite escritura y el nombre del archivo. El método *output()* llamado en la línea 56 escribe los datos PostScript a este

archivo, pero como no espera el sufijo *.ps*, el sufijo se quita primero de la línea 55 y luego el resultado es escrito a *\$base*.

Tras llamar al comando de impresión en la línea 59, depende de la línea 63 el que se elimine el archivo temporal obsoleto. *textbox()* en la línea 68 crea un Nuevo objeto *PostScript::TextBlock* y llama al método *addText*. Éste espera el nombre de la fuente y su tamaño, el valor del espaciado interlineal (*\$leading*) y el texto que se debe fijar.

Para determinar el tamaño de la caja de texto que se generará, se llama a *tb_width()* y *tb_height()* (*tb* significa bloque de texto), definidas más abajo. Mientras que *tb_height* simplemente necesita multiplicar el espacio interlineal por el número de líneas, calcular el espacio horizontal usado es más complicado debido a la variación en la anchura de los caracteres.

Asistente para Medir Fuentes

Afortunadamente hay un módulo llamada *PostScript::Metrics* con una función llamada *stringwidth()* que usa tablas incrustadas de fuentes para resolver estos problemas. La mala noticia es que el módulo nunca ha oído hablar de *Helvetica-iso*. Simplemente quitando el sufijo *-iso* de la línea 96 resolvemos este problema, pero impedimos que funcionen caracteres especiales. Esto nos lleva a otra solución en la línea 101, donde unos pocos caracteres son simplemente reemplazados por la letra *A*. si bien esto no genera un resultado preciso, sí solucionó el problema de nuestro ejemplo.

Reconozco que he tenido que usar unos pocos trucos en esta ocasión, pero mi excusa es que he tenido que encontrar la manera de resolver la incompleta implementación de los módulos *PostScript::**. Es un precio que estoy dispuesto de pagar, pues me permite elegir la libertad de Linux y evitar todos esos arranques superfluos de Windows. ■

INFO

- [1] Como instalar impresoras en Linux: <http://www.linuxprinting.org>
- [2] Shawn Wallace, "Perl Graphics Programming": O'Reilly, 2002
- [3] Más información sobre PostScript: <http://www.mathematik.uni-ulm.de/help/pstut/>