Manual de Groff (Escrito en Groff).

Autor: P.L. Lucas

Índice

1 Manual de groff

Groff es un procesador de textos eclipsado por el uso de LibreOffice, LATEX,... ¿Por qué es interesante este programa? Groff es la base del sistema de documentación de Gnu y las páginas de los manuales se escriben es este formato. Por lo tanto viene instalado de serie en nuestra distribución de Linux favorita. Si ya lo tienes instalado, habrá que darle una oportunidad.

LibreOffice, OpenOffice, Google Docs, Word,... todos son procesadores de texto WYSIWYG (What You See Is What You Get) lo que escribes es lo que al final obtienes al imprimir. Existen otros procesadores de texto cuyo resultado no se ve mientras escribes el texto sino cuando ejecutas un comando específico. La tarea en estos procesadores de texto es más similar a realizar una página web. LATEX es el mejor ejemplo de este último tipo de procesadores. Revistas, libros o tesis son escritas con LATEX a diario y consiguen unos resultados de gran calidad.

¿Por qué usar un procesador de texto no WYSIWYG? Simple, cuando se está redactando un documento corto o sencillo, es más eficiente usar un procesador de textos WYSIWYG. Pero cuando se escribe un gran libro con muchas imágenes o figuras, el proceso de "maquetación", colocar el texto y figuras del libro para que queden visualmente atractivas, puede ser un gran dolor de cabeza. En un procesador de textos no WYSIWYG se definen unas reglas para el texto y las figuras. Si se definen correctamente dichas reglas, no hay que preocuparse más por la maquetación del texto.

Hay un viejo refrán que dice: "Costurera sin dedal cose poco y mal."
Este refrán hace referencia a que hay que tener las herramientas adecuadas para cada tarea. Un procesador de textos WYSIWYG es una herramienta muy útil, pero... ¿Es siempre la más adecuada? LATEX o LibreOffice ocupan un gran tamaño a la hora de instalarlos. Son grandes procesadores de texto con unas capacidades increíbles. Pero a veces puede interesar, por ejemplo, generar una salida en PDF con un script o programa sencillo que se haya realizado. Aquí puede ser muy útil Groff pues permite que se le introduzcan los comandos usando las típicas tuberías de UNIX, cosa que por ejemplo en LATEX no es posible.

El origen de Groff se encuentra en el programa Troff. Por ello si hacemos alguna búsqueda en Internet sobre alguna duda sobre groff, acabaremos leyendo manuales de troff. Groff es la versión de software libre de Troff.

Por supuesto, este manual ha sido escrito usando... Groff.

2 El comando groff y las tuberías

Groff se diseñó para poder ser usado con la tuberías UNIX lo cual le permite ser ampliado con mucha facilidad. Básicamente Groff consiste en una serie de comandos que permiten dar tipo de letra, tamaño, color, posición,... al texto.

Por ejemplo, sea el siguiente fichero en formato groff:

```
Hola mundo.
.br
Esto es un ejemplo.
```

Se puede escribir usando Gedit, Kwrite o vi y lo guardamos en un archivo llamado "ejemplo1.txt". Después ejecutamos el siguiente comando:

```
groff -dpaper=a4 ejemplo1.txt | ps2pdf - ejemplo1.pdf
```

Al final se obtiene un archivo PDF con el siguiente documento:

Hola mundo. Esto es un ejemplo.

Si introdujéramos algún acento, como normalmente usamos UTF-8 para escribir en Linux, groff no los interpretaría correctamente y veríamos unos extraños símbolos en lugar de los acentos y nuestra querida ñ. Aquí es donde se puede empezar a apreciar la potencia de Groff para ser ampliado a través de las tuberías. El comando a ejecutar en nuestra terminal ahora sería:

```
preconv ejemplo1.txt | groff -dpaper=a4 | ps2pdf - ejemplo1.pdf
```

"preconv" es un programa que traduce el UTF-8 en símbolos que Groff puede interpretar. Ahora sí se verían correctamente los acentos y la gran mayoría de los caracteres UTF-8.

Groff no soporta la edición de fórmulas matemáticas. Con el programa "eqn" se pueden introducir estas fórmulas. Por ejemplo, sea el siguiente archivo que vamos a llamar "ejemplo2.txt":

```
.EQ
x={-b +- sqrt{b sup 2 - 4ac}} over 2a
.EN
```

Lo procesamos con:

```
preconv ejemplo2.txt | eqn | groff -dpaper=a4 | ps2pdf - ejemplo2.pdf
```

Al final se obtendrá:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

La sintaxis de "eqn" es muy similar a la que usa el programa LibreOffice para representar ecuaciones.

De forma similar se pueden generar tablas con el programa "tbl":

```
preconv archivo.txt | eqn | tbl | groff -dpaper=a4 | ps2pdf - archivo.pdf
```

3 Conceptos básicos

En el apartado anterior ya se ha visto la forma de procesar los archivos y generar un PDF. Básicamente es abrir una archivo con nuestro editor de textos favorito, como puede ser Gedit o Kwrite y escribir allí nuestra obra maestra.

Por ejemplo, en un archivo se puede guardar el siguiente texto:

```
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lantejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda.

El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con sus pantuflos de lo mesmo, y los días de entresemana se honraba con su vellorí de lo más fino.

Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera.
```

Como se puede apreciar el texto es bastante simple de introducir. Los párrafos los separamos introduciendo una línea en blanco. Por ahora no se ha hecho nada en especial. Al procesarlo usando los comandos del apartado anterior y ver el PDF resultante, hay que fijarse que se han puesto unos márgenes de página bastante incómodos. Más adelante se verá la forma de modificarlos.

Vamos a jugar con los tipos de letra. Lo primero es el estilo, se usará ".ft" para cambiar entre cursiva, negrita o regular.

Importante: Todos los comandos de groff que comiencen por un punto se deberán escribir al comienzo de la línea para que puedan ser ejecutados. Es decir, el punto será el primer carácter que se aparecerá al comienzo de la línea.

```
Este texto es normal.
.ft B
Texto en negrita.
.ft
El texto continua normal.
\" Esto es un comentario y se ignora
```

Que nos generará la siguiente salida:

Este texto es normal. **Texto en negrita.** El texto continua normal.

Usando \" se pueden introducir comentarios que el procesador de textos ignora. En el ejemplo también se aprecia que los saltos de línea no afectan a la continuidad del párrafo. Aunque al introducir el comando ".ft" se genera una nueva línea, el texto aparece a continuación del anterior. Si nos fijamos en el siguiente ejemplo:

```
Este texto es normal.
```

```
.ft B
Texto en negrita en otra línea.
.ft
El texto continua normal.
```

Generará:

Este texto es normal.

Texto en negrita en otra línea. El texto continua normal.

Si se necesita comenzar una línea con un punto, se deberá poner un \& al principio de la línea. Por ejemplo:

```
\&.ft
```

3.1 Jugando con los tipos de letra

Como ya se ha visto el comando ".ft" cambia el estilo del texto y admite los siguientes argumentos: B (negrita), I (Cursiva) y R (normal). Se pueden mezclar. Por ejemplo, negrita y cursiva:

```
.ft BI
Texto en negrita y cursiva
.ft
```

El comando ".ft" sin argumentos, lleva el texto al estado anterior a la modificación.

Para cambiar el tipo de letra y no el estilo se usará el comando ".fam". Entre los tipos de letra que vienen por defecto en groff se encuentran: T (Times), H (Helvética), C (Courier), ZD (Dingbats), S (símbolos griegos y matemáticos), N (New Century Schoolbook),...

```
.fam H
Texto en Helvética
.fam
Se vuelve al tipo de letra anterior.
```

Algunos tipos de letra con su código son:

A	Avant Garde Book	Hola mundo
BM	Bookman	Hola mundo
C	Courier	Hola mundo
H	Helvetica:	Hola mundo
HN	Helvetica Narrow	Hola mundo
N	New Century Shcoolbook	Hola mundo
P	Palatino	Hola mundo
T	Times Roman	Hola mundo

Mención a parte merecen los siguientes tipos de letra:

ZD	Zapf Dingbats	✐✁✉☛☞
ZDR	Zapf Dingbats Reversed	✐✁✉☚☜
ZCMI	Zapf Chancery	*Hola mundo*
S	Symbol	ϒξψζ
SS	Symbol Slanted	*ϒξψζ*

Los tipos de letra ZD, ZDR, ZCMI, S y SS no se cambian con el comando ".fam", sino con ".ft":

```
.ft ZCMI
Hola mundo
.ft
.ft S
\N'200'\N'120'\N'121'\N'122'
.ft
.ft SS
\N'200'\N'120'\N'121'\N'122'
.ft
```

El tipo de letra Zapf Dingbats (ZD y ZDR) necesita usar el comando:

```
\N'Número'
```

para poder introducir los caracteres. El número es un número entre 0 y 255. En los apéndices del final de este documento se puede encontrar una tabla con el listado de símbolos. Por ejemplo, para escribir el símbolo ☞ se usará:

```
.ft ZD
\N'43'
.ft
```

Se pueden escribir varios símbolos seguidos:

```
.ft ZDR
\N'50'\N'40'\N'41'\N'42'\N'43'
.ft
```

Otra forma de cambiar el tipo de letra es con \f[]. Entre los corchetes se pondrá el tipo de letra. Se usará con los corchetes vacíos para volver al tipo de letra anterior. Por ejemplo:

```
\f[B]a\f[]\f[I]f\f[]a\f[ZDR]\N'50'\f[]
```

Devolverá: **a***f*a✁

La forma \f[] para cambiar los tipos de letra se usará dentro de las líneas de texto para no introducir espacios entre las letras.

Para cambiar el tamaño de la letra se usará ".ps" seguido de un número que representa el tamaño de la letra:

```
.ps 10
Hola
```

```
.ps \" Se vuelve al tamaño anterior
Adios
.ps 12
Hola
.ps \" Se vuelve al tamaño anterior
Adios
.ps 14
Hola
.ps \" Se vuelve al tamaño anterior
Adios
.ps 20
Hola
.ps \" Se vuelve al tamaño anterior
Adios
```

Esto da como resultado:

Hola Adios Hola Adios Hola Adios Hola Adios

El comando ".ps" sin argumentos lleva al texto a su tamaño anterior.

4 Márgenes y tamaño de la página

Si habéis estado haciendo estas pruebas veréis que los márgenes que usa son muy estrechos, los podemos cambiar usando los siguientes comandos:

`.po n` fija el margen izquierdo a n unidades
`.ll n` fija la longitud de la línea en n unidades
`.pl n` fila la longitud de la página en n unidades

Las unidades que se pueden usar son:

u	Unidad interna.
c	Centímetros.
i	Pulgadas. Son 432 unidades internas en troff.
v	Espacio de línea vertical.
p	Punto (1/72 de pulgada). Son 6 unidades internas en troff.
P	Pica (1/6 de pulgada). Son 72 unidades internas.
m	Unidad m (tamaño de una letra 'm' en el tipo y estilo actual).
n	Unidad n (tamaño de una letra 'n' en el tipo y estilo actual).

Para un A4 en comando .pl debería ser:

```
.pl 29.7c
```

En este documento se han establecido las siguientes unidades:

```
.po 1.5c
.ll 18c
.pl 29.7c
```

El tamaño de la página se puede controlar desde el comando que se lanza groff para procesar el documento. El argumento -dpaper indica el tamaño. Para un A4 se usaría:

```
preconv ejemplo2.txt | eqn | groff -dpaper=a4 | ps2pdf - ejemplo2.pdf
```

Si se quiere la hoja apaisada le añadirá una "l" al final del tamaño. Para un A4 apaisado:

```
preconv ejemplo2.txt | eqn | groff -dpaper=a4l | ps2pdf - ejemplo2.pdf
```

Al definir los márgenes de la página, no se ha hablado nada del margen superior e inferior. Esto se debe a que no se pueden modificar directamente. Para controlar estos márgenes se usarán las *trampas*. Una trampa es una macro que se ejecuta al llegar a una determinada posición en el texto. Lo que se hará es poner trampas al principio y al final de la página que vayan cambiado la posición del texto o introduzcan una nueva página. También se pueden usar las trampas para introducir los encabezados y pies de página.

Más adelante de este documento se verá cómo usar las trampas para definir el tamaño del margen superior y el inferior.

4.1 Un documento sencillo

Con lo que ya sabemos ya se podría iniciar la edición de un documento sencillo, hay que introducir lo siguiente al principio del documento para que nos controle los márgenes:

```
\" ------------------------------------------------------
\" Se introducen las macros de inicio
\" ------------------------------------------------------
\" Encabezado y pie de página
.de Encabezado
.   ev 1
.   fam N
.   ps 10
.   ft R
\" Descomentar la siguiente línea para poner un encabezado al documento:
\" .tl ''\v'|0.5c'Encabezado del documento''
.   ft
.   ps
.   fam
.   ev 0
..
.de PiePagina
.   ev 1
.   mk
.   fam N
.   ps 10
.   ft R
.   tl ''\v'|28c'Página %''
.   ft
.   ps
.   fam
.   rt
.   ev 0
.   di huerfano \" Se captura la palabra que haya quedado huérfana
.   br
.   di
```

```
.   bp \" Se cambia de página y se escribe el huérfano
.   huerfano
..

.wh 0c Encabezado
.wh -2.5c PiePagina


\" -----------------------------------------------------
\" Márgenes de página
.po 1.5c        \" Margen izquierdo
.ll 18c         \" Longitud de la línea
.pl 29.7c       \" Longitud de la página
\" .ls 1           \" Interlineado en líneas
.vs 16          \" Interlineado en puntos
.hy 1           \" Separar las palabras con guiones al final de la línea


\" -----------------------------------------------------
\" Tipo de letra del documento
\" Los tipos por defecto son:
\" T (Times), H (Helvética), C (Courier), N (New Century Schoolbook)
\" Los modificadores son: Times normal (R), cursiva (I), negrita (B),
\" ZD (Dingbats), S (símbolos griegos y matemáticos)
.fam N
.ft R
\" Tamaño por defecto
.ps 11

\" -----------------------------------------------------
\" Títulos y subtítulos
.nr tituloNivel1 0 1
.nr tituloNivel2 0 1

.de1 Titulo1
.fam H
.ft B
.ps 14
\" .nr tituloNivel1 +1
\\n+[tituloNivel1] \\$1
.ps
.ft
.fam
\\R'tituloNivel2 0'
..

.de1 Titulo2
.fam H
.ft B
.ps 12
\\n[tituloNivel1].\\n+[tituloNivel2] \\$1
.ps
.ft
.fam
..

\" Se restaura la posición del documento
.rt 1c
\" -----------------------------------------------------
\" Cuerpo del documento
\" A partir de aquí va tu obra maestra
```

Se han definido una serie de macros para introducir los encabezados y pies de página. También se definen las macros ".Titulo1" y "Titulo2". Estas macros permiten introducir títulos similares a los de este documento. Por ejemplo:

```
.Titulo1 "Introducción"
Aquí va la introducción..
.Titulo2 "Historia"
Aquí va la historia
.Titulo2 "Siguiente apartado"
...
```

La salida sería algo así como:

1. Introducción
Aquí va la introducción
1.1 Historia
Aquí va la historia
1.2 Siguiente apartado
...

Evidentemente las macros se pueden modificar para ajustar el documento a nuestros gustos.

Existen paquetes de macros más avanzados como "ms", cuya documentación se puede consultar en el manual de Groff, que permiten títulos, índices, enumeraciones,... En estos paquetes modificaremos las macros para que el texto se adapte a nuestras necesidades. La filosofía de trabajo cambia con respecto a un procesador de texto WYSIWYG en el que el usuario va ajustando la posición de las figuras y tablas a manos. Ahora lo que se hará será definir unas políticas de colocación de las tablas, imágenes, figuras y el texto. Si estas políticas están bien definidas, no nos tendremos que preocupar de colocar a mano tal o cual figura y si, posteriormente, se modifica algo en el texto anterior a la figura las macros deberán ocuparse de que el texto y las figuras no se descabalen.

5 Formato de los párrafos

Groff permite controlar la alineación, interlineado y sangrías de los párrafos de forma sencilla. Veamos cómo:

5.1 Alineación del texto

A la hora de manejar los párrafos de texto, a veces interesará ajustar el texto a la izquierda, a la derecha o justificarlo. Para esta tarea se usa el comando ".ad":

`.ad l` Ajusta al margen izquierdo.
`.ad r` Ajusta al margen derecho.
`.ad c` Centra el párrafo.
`.ad b` Justifica el párrafo (por defecto).

Por ejemplo:

```
.ad c
Texto centrado
```

```
.ad b
Texto normal
```

Produce la siguiente salida:

Texto centrado

Texto normal

5.2 Interlineado

Para controlar la separación entre las líneas de un párrafo tenemos los siguientes comandos:

`.ls N` Fija el espaciado entre líneas a N líneas. No se permiten decimales
`.vs N` Fija el espaciado entre líneas en N puntos

Con el comando ".ls" no se pueden fijar interlineados de 1.5 líneas de separación. En ese sentido ".vs" permite un control mucho más preciso.

Por ejemplo:

```
Texto de dos líneas muy separadas
.br
La otra línea

.vs
Se vuelve al interlineado anterior
.br
Se vuelve al interlineado anterior
```

Esto produce la siguiente salida:

Texto de dos líneas muy separadas

La otra línea

Se vuelve al interlineado anterior
Se vuelve al interlineado anterior

5.3 Sangrado de párrafos

Para sangrar los párrafos, se tienen los siguientes comandos:

`.ti N` Sangra la primera línea del siguiente párrafo en N unidades.
`.in N` Sangra todas las líneas de los párrafos siguientes N unidades.

Por ejemplo:

```
.ti 3c \" Se sangra en 3cm
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho
tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua,
rocín flaco y galgo corredor. Una olla de algo más vaca que carnero,
salpicón las más noches, duelos y quebrantos los sábados, lantejas los
viernes, algún palomino de añadidura los domingos, consumían las tres
partes de su hacienda.

\" Este párrafo no se sangra
El resto della concluían sayo de velarte, calzas de velludo para las
fiestas, con sus pantuflos de lo mesmo, y los días de entresemana se
honraba con su vellorí de lo más fino.

.ti 1c
Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no
llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el
rocín como tomaba la podadera.
```

Esto genera la siguiente salida:

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lantejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda.

El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con sus pantuflos de lo mesmo, y los días de entresemana se honraba con su vellorí de lo más fino.

Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera.

Veamos ahora el siguiente ejemplo:

```
.in 3c \" Se sangra en 3cm
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho
tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua,
rocín flaco y galgo corredor. Una olla de algo más vaca que carnero,
salpicón las más noches, duelos y quebrantos los sábados, lantejas los
viernes, algún palomino de añadidura los domingos, consumían las tres
partes de su hacienda.

El resto della concluían sayo de velarte, calzas de velludo para las
fiestas, con sus pantuflos de lo mesmo, y los días de entresemana se
honraba con su vellorí de lo más fino.

.in -1c
```

```
Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no
llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el
rocín como tomaba la podadera.
```

Produce la salida:

> En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lantejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda.
>
> El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con sus pantuflos de lo mesmo, y los días de entresemana se honraba con su vellorí de lo más fino.
>
> Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera.

Para que el texto vuelva a su posición inicial, hay que restar lo que hayamos sangrado:

```
.in 3c \" Se sangra el texto 3cm
Texto muy largo

.in -3c \" Se le restan 3cm al sangrado para volver a la posición 0cm
Volvemos al sangrado normal
```

5.4 Separación automática de palabras

Si nos fijamos en el presente documento, al final de los párrafos, algunas palabras se dividen usando guiones. Este proceso lo realiza Groff de forma automática. Para controlarlo usamos los siguientes comandos:

`.hy N` Activa la separación de palabras usando guiones.

El parámetro N puede tener los siguientes valores:

0 Desactiva los guiones
1 Activa guiones
2 Impide la separación de la última palabra en una página o columna
4 Impide la separación de los dos últimos caracteres de una palabra
8 Impide la separación de los dos primeros caracteres de una palabra

Estos valores se pueden sumar para tener activadas dos opciones. Por ejemplo, un valor 12 no separa ni los dos últimos ni los dos primeros caracteres.

Con el comando .hw se puede especificar la separación específica de una palabra. Por ejemplo:

```
.hw soft-wa-re
```

Con el comando .hla se define el idioma para realizar la separación:

```
.hla us
```

Por desgracia el español (es) no está entre los idiomas por defecto. Afortunadamente se pueden usar las tuberías y un poco de programación para resolver el problema. Debemos copiar el siguiente código python3 en un archivo llamado "hyphen-es.py" (es importante respetar las tabulaciones):

```
#!/usr/bin/python3
import sys

listado = []

for linea in sys.stdin:
    sys.stdout.write(linea)
    for palabra in linea.split():
        if palabra.startswith('.') or len(palabra)<4:
            continue
        p=palabra.lower()
        n=len(p)
        salida = ""
        vocalPrevia = False
        while n>0:
            n-=1
            ch = p[n]
            if not ch.isalnum() or ch.isdigit():
                salida = ""
                break
            if ch in ['a', 'e', 'i', 'o', 'u', 'á', 'é', 'í', 'ó', 'ú']:
                if vocalPrevia and not ch in ['i', 'o', 'u', 'í', 'ó', 'ú']:
                    salida = '-{0}'.format(salida)
                vocalPrevia = True
                salida = '{0}{1}'.format(ch, salida)
            else:
                if vocalPrevia:
                    ch1 = ch
                    if n>0:
                        ch1 = p[n-1] + ch
                    salida = '{0}{1}'.format(ch, salida)
                     if not ch1 in ['bl', 'br', 'ch', 'cl', 'cr', 'dr', 'fl', 'fr', 'gl', 'gr', 'll',
'pl', 'pr', 'tr']:
                        salida = '-{0}'.format(salida)
                        vocalPrevia = False
                    else:
                        vocalPrevia = True
                else:
                    salida = '{0}{1}'.format(ch, salida)
                    vocalPrevia = False
        if salida.startswith('-'):
            salida=salida[1:]
        if len(salida)>0 and salida not in listado:
            listado.append(salida)

fout = open("hyphen-es.txt", "w")
for p in listado:
    fout.write('.hw {0}\n'.format(p))
fout.close()
```

Para procesar nuestro fichero, ahora se debe usar la cadena de comandos:

```
soelim archivo.txt | python3 division-es.py
soelim archivo.txt | preconv | eqn | tbl | groff -dpaper=a4 | ps2pdf - ma-
nual_groff.pdf
```

e incluir al principio de nuestro archivo de texto las líneas:

```
.hla es
.so hyphen-es.txt
```

El comando .so sirve para insertar un documento dentro de nuestro texto. "hyphen-es.txt" es la separación de las palabras del documento.

Este programa hace una separación de palabras muy simple, pero funcional.

5.5 Ruptura de líneas y páginas

Para cambiar a una nueva línea, se puede dejar una línea en blanco o se puede usar el comando:

```
.br
```

A veces interesa cambiar a una nueva página, para ello se usará el comando:

```
.bp
```

6 Insertando imágenes

Tarde o temprano será necesario introducir una imagen o dibujo en nuestro documento. Para ello se usa el siguiente comando:

```
.PSPIC [Posicion] archivo [anchura [altura]]
```

Donde [Posición] puede valer:

- `-L` Alinea la imagen a la izquierda.
- `-R` Alinea la imagen a la derecha.
- `-C` Coloca la imagen en el centro (es la opción por defecto).
- `-I n` Provoca el sangrado de la imagen en n unidades.

Los campos anchura y altura son opcionales.

Por ejemplo:

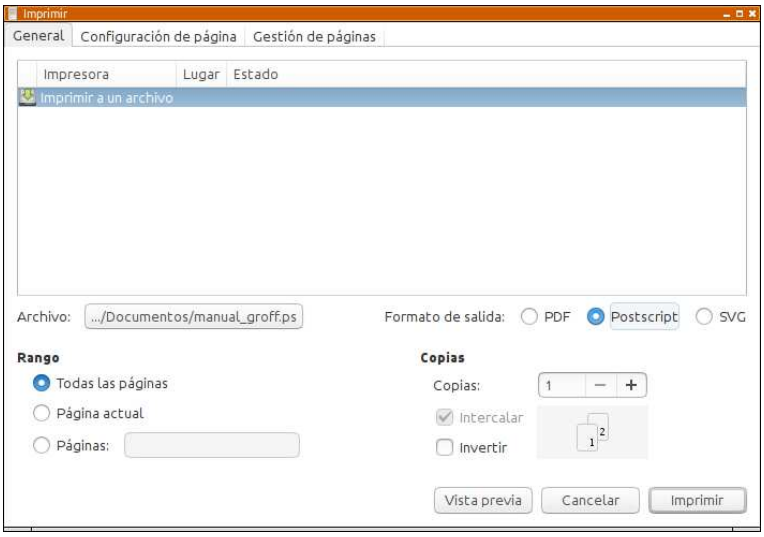
```
.PSPIC -L imagen.eps 3c
```

Inserta el archivo "imagen.eps" a la izquierda con una anchura de 3cm.

Al igual que LATEX, groff trabaja con figuras en formato EPS (Encasulated PostScript), por lo que nuestras imágenes o figuras se deberán traducir antes al formato EPS. Esto se puede hacer de varias formas. La más simple es usar el programa "convert" del paquete ImageMagik:

```
convert archivo archivo.eps
```

Otra opción es imprimir nuestra imagen directamente en Postscript. Cuando vamos a imprimir un documento, los programas de Linux permiten que se imprima en un archivo en formato Postscript:



Después se puede usar el programa "ps2eps" para traducir el Postscript en EPS. "ps2eps" sólo vale para documentos que ocupen una hoja (que es lo típico a la hora de hacer imágenes para incluir en un documento).

Otro formato permitido es el EPSI, que es una versión del formato EPS que incrusta una imagen en mapa de bits de la figura para que lo representen rápidamente los procesadores de texto. Se puede usar el comando "ps2epsi" para conseguirlo.

Al insertar una imagen no podemos colocar texto al rededor de ella. Usando macros y trampas, se puede resolver este problema.

7 Almacenando y recuperando posiciones

A veces es necesario almacenar la posición actual en la página, hacer una serie de operaciones y volver a esta posición. Por posición se entiende la distancia que hay desde la parte superior de la página hasta la posición actual.

La acción de almacenar una posición se realiza con:

```
.mk
```

Para recuperar la posición que se ha almacenado se usa el comando:

```
.rt
```

Es recomendable usar una variable para almacenar la posición. Por ejemplo:

```
.mk figura
```

Almacenará en la variable figura el valor de la posición.

Un ejemplo, supongamos que se quiere colocar el texto en dos columnas. Para ello se usarán los comandos ".in" y ".ll" que ya se han presentado anteriormente:

```
.ll 8c \" Se establede la anchura de la primera columna a 8cm
.mk \" Se almacena la posición actual
...Aquí va el texto de la primera columna...
```

```
.rt \" Se vuelve a la posición anterior
.ll \" Se restaura el ancho de la página
.in 9c \" Se indica el margen de la siguiente columna
...Aquí va el texto de la segunda columna...
.in -9c
```

Esto da como resultado:

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lantejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda.

El resto della concluían sayo de velarte, calzas de velludo para las fiestas, con sus pantuflos de lo mesmo, y los días de entresemana se honraba con su vellorí de lo más fino. Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera.

También se podría hacer que el texto fluyera al rededor de una imagen o de una tabla. El problema es que debemos conocer la extensión de los textos para saber cuándo vamos a romper (ejecutar ".rt") la columna. Este problema se resolverá usando un mecanismo llamado trampa que provoca la ejecución de una macro cuando el texto alcanza una determinada posición.

7.1 Posicionamiento vertical y horizontal

Otra forma de usar el comando ".rt" es indicar la posición vertical a la que se desea ir. Esto se hace pasando un argumento al comando ".rt":

```
.rt 1c \" Se coloca la posición a 1cm del borde superior
```

Si se ejecuta ".rt" de esta forma, no es necesario llamar a un comando ".mk" anterior.

Se puede colocar el texto en una posición dada con lo anterior o usando \v. Por ejemplo:

```
\v'|0.5c' Hola mundo
```

Escribirá a 0.5cm de la parte superior el texto que tenga a continuación. Si se escribe un párrafo nuevo se vuelve a la posición anterior.

Hay dos diferencias entre usar ".rt" y "\v" para posicionar el texto:

① ".rt" no vuelve a la posición inicial después del primer párrafo. Con "rt" los siguientes párrafos se muestran a partir de la nueva posición indicada.
② "\v" no activa las trampas al hacer un cambio de posición. Se verá más adelante al explicar las trampas.

Al usar "\v" se ha colocado el símbolo "|" en el argumento. Esto significa que la posición se hará respecto a la parte superior de la página. Si se quita este símbolo, la colocación se hará respecto de la posición actual. Así:

```
Hola\v'0.1c' mundo
```

Mostrará la siguiente salida:

Hola mundo

La posición puede ser tanto positiva como negativa:

```
Hola\v'-0.1c' mundo
```

Hola mundo

Al igual que se posiciona con "\v" verticalmente se puede posicionar de forma similar con "\h". Por ejemplo:

```
Hola\h'-1c' mundo
```

Se produce que la palabra "mundo" se sobrescriba sobre la palabra "Hola".

Homundo

También se puede usar el símbolo "|" para especificar la posición respecto el principio del párrafo:

```
\h'|0.5c' Hola
```

Jugando con la posición horizontal y vertical, se puede conseguir, por ejemplo, el símbolo de TEX:

```
T\h'-.1667m'\v'.224m'E\v'-.224m'\h'-.125m'X
```

8 Macros

Con lo que se ha visto hasta ahora ya se podría hacer un documento que luciera bastante bien. Pero hay formas de ahorrarnos escribir y dar formato al texto de forma más eficiente.

Una macro es una función que podemos definir y llamar en cualquier momento. Por ejemplo, supongamos que tenemos que escribir el mismo texto muy largo varias veces. Se puede definir una macro con dicho texto y cada vez que llamemos a la macro se escribirá el texto. Por ejemplo:

```
.de textoLargo \" Se define la macro textoLargo
Este es un texto muy largo
..

.textoLargo \" Se ejecuta la macro
```

Al ejecutar la macro se obtiene la salida:

Este es un texto muy largo

Cada vez que se llame a la macro ".textoLargo" se escribirá el texto correspondiente.

Como se puede ver en el ejemplo, la macro comienza con ".de" y termina con ".."

El nombre de la macro puede admitir cualquier carácter salvo:

- Caracteres en blanco (espacios, tabuladores y saltos de línea)
- El caracter borrar y el carácter con código 0x01

- Cualquiera de los caracteres siguientes: 0x00, 0x0B, 0x0D-0x1F, 0x80-0x9F.

Las macros pueden admitir argumentos al igual que lo hacen los comandos en la terminal de Linux:

```
.de Argumentos
Primer argumento: \\$1
.br
Segundo argumento:  \\$2
..

.Argumentos "El Primero" Segundo
```

Esto devuelve la salida:

Primer argumento: El Primero
Segundo argumento: Segundo

En el ejemplo se puede ver que después de llamar a la macro se colocan los argumentos. A los argumentos se accede usando \\$1 para el primer argumento, \\$2 para el segundo, \\$3 para el tercero,... \\$0 es el nombre de la macro que se está invocando. Las comillas permiten agrupar varias palabras solamente en un argumento:

```
.Argumentos Primero Segundo
```

Devuelve:

Primer argumento: Primero
Segundo argumento: Segundo

Pero:

```
.Argumentos "Primero Segundo"
```

Devuelve:

Primer argumento: Primero Segundo
Segundo argumento:

En el caso de querer introducir unas comillas en el argumento se puede usar \N[34] (ver tablas de caracteres en los apéndices).

```
.Argumentos "Primero \N[34] Segundo"
```

Si, por ejemplo se están escribiendo unas cartas, todas con el mismo contenido, pero cambiando a la persona:

```
.de Carta
\\$1 le informamos de que su pedido está listo para ser recogido y
bla, bla, bla,...
..

.Carta "Sr. Pepe Pérez"
.Carta "Sra. Pepita Pérez"
```

De igual forma se pueden usar para definir estilos, cambios en el tipo de letra, para insertar un carácter especial,...

Otros parámetros interesantes son:

`\n[.$]` Es el número de argumentos que se le han pasado a la macro.

`\$*` Concatena todos los argumentos separados por espacios.

`\$@` Concatena todos los argumentos separados por comillas.

`.shift` Elimina el primer argumento que se haya pasado. Si se le pasa un número "n" como argumento, elimina los "n" primeros ardumentos.

Por ejemplo:

```
.de ejemplo
\\\\\\\\n[.$] = \\\\n[.$]
\\\\\\\\$* = >\\\\$*<
\\\\\\\\$@ = >\\\\$@<
.   shift
\\\\\\\\n[.$] = \\\\n[.$]
\\\\\\\\$@ = >\\\\$@<
..
.ejemplo Hola mundo "Adios mundo"
```

Genera la salida:

```
\n[.$] = 3
\$* = >Hola mundo Adios mundo<
\$@ = >"Hola" "mundo" "Adios mundo"<
\n[.$] = 2
\$@ = >"mundo" "Adios mundo"<
```

9 Registros

A las variables numéricas en Groff se las denomina *registros*. Al igual que en matemáticas, una variable es una letra (o un identificador) en la que podemos almacenar un valor numérico. Los identificadores son los mismos que se pueden usar para nombrar un macro.

Para definir un registro se puede proceder de las siguientes formas, o bien se usa el comando .nr:

```
.nr identificador valor
```

Que asocia al identificador el valor dado. Por ejemplo:

```
.nr a 2
```

Asocia al registro `a` el valor 2.

Para ver el valor que tiene el registro se usa `\n[identificador]`. Por ejemplo:

```
.nr a 2
\n[a]
```

Escribirá un 2 como salida.

Si se ponen paréntesis rodeando cada operación, se pueden hacer cuentas a la hora de asignar valo-

res:

```
.nr a (3+(3*4))
\n[a]
```

Devolverá 15.

Si no se ponen los paréntesis, el resultado es 24:

```
.nr a 3+3*4
\n[a]
```

Esto se debe a que Groff no respeta el orden de las operaciones matemáticas. 3+3*4 debería resultar 12, pero Groff lo interpreta como ¡¡ 3+3*4 = 6*4 = 24 !!

A un registro se le puede sumar el valor de otro registro u otras operaciones matemáticas. Por ejemplo:

```
.nr a 3
.nr b 2
.nr a \n[a]+\n[b]+3
\n[a]
```

Da como resultado 8.

Existe una forma especial de sumar o restar un valor a un registro:

```
.nr a +1
```

Sumaría 1 al valor de a. También vale con valores negativos:

```
.nr a -5
```

Restaría 5 al valor de a.

Otro ejemplo:

```
.nr a 2
.nr a +4
\n[a]
```

Da como resultado 4.

Esta notación plantea un problema, pues en el caso de que se quiera asignar un registro a otro cambiado de signo, lo que haría es restar el valor al registro:

```
.nr a 5
.nr b 1
.nr a -\n[b]
\n[a]
.nr a (-\n[b]) \" Los paréntesis permiten asignar el valor negativo
\n[a]
```

Devolvería 4 y -1 respectivamente.

Se podrían usar los registros para crear enumeraciones. Por ejemplo:

```
.nr a 1
\n[a] Pan
```

```
.br
.nr a +1
\n[a] Leche
.br
.nr a +1
\n[a] Fruta
```

Da como resultado:

1 Pan
2 Leche
3 Fruta

En el caso de querer eliminar una variable se usará .rr:

```
.rr a
```

9.1 Registros especiales

Existen algunos registros especiales que pueden ser de ayuda en determinados casos, como son:

`\n[%]` Devuelve el número de la página actual.
`\n[.F]` Devuelve el nombre del archivo actual.
Los registros `\n[hours]`, `\n[minutes]` y `\n[seconds]` devuelven la hora, minutos y segundos actuales.
`\n[dw]` Día de la semana (1-7)
`\n[dy]` Día del mes (1-31)
`\n[mo]` Mes actual (1-12)
`\n[year]` Año actual.

Existen más registros especiales que se pueden consultar en el manual de referencia de groff.

9.2 Registros y macros

Para entender el uso de los registros dentro de las macros, se va a investigar el siguiente ejemplo:

```
.nr a 3  \" Se define el registro a antes de la macro
\n[b]    \" Como el registro b no está definido, devuelve 0
.de ejemplo     \" Se define la macro ejemplo
.nr a 2     \" Se le da al registro a el valor 2
.nr b 4     \" Se le da al registro b el valor 4
\n[a]   \" Devuelve 3
\\n[a]  \" Devuelve 2
\n[b]   \" Devuelve 0
\\n[b]  \" Devuelve 4
..
\" Se ejecuta la macro ejemplo
.ejemplo
\n[a]   \" Devuelve 2
```

```
\n[b]   \" Devuelve 4
```

Como se puede ver en el ejemplo, para trabajar con los registros en macros se tienen dos operadores, por un lado está el operador \n[] que devuelve el valor del registro antes de entrar en la macro. Por otro lado se tiene \\n[] que devuelve el valor del registro con las operaciones que haya recibido dentro de la macro.

Si se ven los valores que se devuelven después de ejecutar la macro, se puede ver que dentro de la macro se pueden definir nuevos registros y que los registros tendrán, al final, el valor que la macro les haya dejado.

Por ejemplo, para definir una enumeración se podrían definir las siguientes funciones:

```
.de Enumeracion
.nr enum_numero 1
..

.de item
.br
\\n[enum_numero]
.nr enum_numero +1
..

.de finEnumeracion
.rr enum_numero
..

.Enumeracion
.item
Pan
.item
Leche
.item
Fruta
.finEnumeracion
```

Se obtendrá como salida:

1 Pan
2 Leche
3 Fruta

10 Trampas

A veces es interesante que cuando el texto llegue a una determinada posición, se ejecute una macro determinada. Por ejemplo, si el texto llega al margen inferior de la página, se debe escribir el número de página al final de la página y pasar a escribir a una página nueva. Otros ejemplos de uso de las trampas son crear columnas de texto, figuras flotantes, encabezados y pies de página,...

A este mecanismo de definir una posición y que una macro se ejecute al llegar a dicha posición o se den unas ciertas condiciones, se le denomina *trampa*.

Existen varios tipos de trampa unas se aplican a características de texto y otras sobre el propio archivo que se está leyendo. Para obtener una salida de Groff se le debe pasar un archivo de texto, pues se pueden definir trampas sobre ese archivo de texto:

Los tipos de trampa que se aplican al archivo de texto son:

- ❋ Trampas de línea en blanco: Se ejecutan cuando se encuentra una línea en blanco en el archivo de texto que se está procesando.
- ❋ Trampas de espacio en blanco al principio: Se ejecutan cuando se encuentran espacios en blanco al principio de la línea de texto que se está leyendo.
- ❋ Trampas de líneas de texto: Se van a leer n líneas de texto del archivo de entrada antes de ejecutar la macro.
- ❋ Trampas de fin de archivo: Se ejecutan al llegar al final del archivo que se está procesando.

Los tipos de trampa que se aplican a la representación del texto son:

- ❋ Trampas de posición en la página: Se ejecutan cuando el texto llega a una determinada posición en la página.
- ❋ Trampas de redirección (Diversion traps): Es idéntica a la anterior pero se aplica a las redirecciones de texto.

Hasta ahora no se ha hablado de las redirecciones de texto. Groff permite que en lugar de escribir el texto en la página el texto se redirija a la memoria del ordenador y desde allí obtener propiedades de cómo quedaría el texto al escribirlo en la página. Por ejemplo, si queremos recuadrar un texto hay que saber lo que ocupa el texto para calcular el tamaño del recuadro que hay que dibujar alrededor.

10.1 Trampas de posición en la página

Estas trampas se ejecutan cuando el texto llega a una determinada posición en la página.

Las trampas sólo se pueden definir para posiciones verticales dentro de la página. Nunca para posiciones horizontales.

Para crear una trampa se usará la siguiente sintaxis:

```
.wh posición nombre_de_la_macro
```

Por ejemplo:

```
.de PiePagina
\v"|24c"Pie de página
.bp
..
.wh 23c PiePagina
```

Define una trampa que se ejecuta cuando el texto alcanza una posición que esté a 23cm desde la parte superior de la página. En este caso ejecuta la macro PiePagina que escribe "Pie de página" en una posición a 24cm de la cabecera de la página y salta a una nueva página.

Si se coloca un número negativo la posición se cuenta desde la parte inferior de la página:

```
.wh -2.5c PiePagina
```

La macro "PiePagina" se ejecuta a 2.5cm de la parte inferior de la página.

Las trampas se usan para definir los márgenes superior e inferior de la página. El siguiente ejemplo está extraído del manual de Groff y se usa para poner los márgenes inferior y superior a una pulgada de la parte inferior y una pulgada de la parte superior respectivamente:

```
.de hd
' sp .5i
. tl 'Title''date'      \" Page header
' sp .3i
..
.
.de fo
' sp 1v
. tl ''%''              \" Page footer
' bp
..
.
.wh 0 hd                \" trap at top of the page
.wh -1i fo              \" trap one inch from bottom
```

Si intentamos usar el código anterior para definir los márgenes superiores e inferiores, nos encontraremos que nos deja una palabra huérfana al llegar a la posición que le hemos definido y salta a la página siguiente. Esto se debe a que Groff primero escribe y después de escribir comprueba si hay macros para ejecutar en esa posición vertical. Por ello siempre se nos quedará una palabra huérfana. Para evitar estos problemas se deberán usar las redirecciones de texto ("divert") que se explicarán más adelante.

Para eliminar una trampa se usará:

```
.ch nombre_macro
```

Eliminará la trampa que ejecute la macro "nombre_macro".

Este comando también se puede usar para cambiar de posición una trampa, por ejemplo:

```
.ch PiePagina 24c
```

Es lo mismo que:

```
.ch PiePagina
.wh 24c PiePagina
```

A veces es necesario definir trampas a partir de la posición actual. Por ejemplo, que se ejecute la trampa a 3cm de la posición actual. Estas operaciones se verán más adelante.

Se puede definir más de una trampa para una posición dada.

10.2 Trampas de espacio en blanco

Aquí hay dos tipos, las que tienen espacios en blanco al principio de la línea y las que se activan al encontrar una línea en blanco.

10.2.1 Trampas de espacio en blanco al principio de la línea

Estas trampas se ejecutan con:

```
.lsm macro
```

Cuando se encuentren espacios en blanco al principio de una línea se disparará la macro indicada. Estos espacios se eliminarán. El número de espacios en blanco eliminados se almacenan en el registro lsn (\n[lsn]) y en lss (\n[lss]) se almacena el espacio horizontal que ocuparían los espacios en blanco si no se hubiesen eliminado.

10.2.2 Trampas de línea en blanco

Esta trampa se ejecuta cuando se encuentra una línea en blanco:

```
.blm trampa
```

10.3 Trampas sobre el archivo entrada

Además de las trampas de espacio en blanco, se pueden establecer otras trampas sobre el archivo que se está leyendo.

10.3.1 Trampas de líneas leídas

Se puede lanzar una trampa que se ejecute después de haber leído un número n de líneas del archivo de entrada. Estas macros se establecen con el comando:

```
.it n macro
```

Donde n es el número de líneas que se van a leer antes de lanzar la macro "macro".

10.3.2 Trampa de fin de archivo

Esta trampa se ejecuta cuando se alcanza el final del archivo de entrada. Se activa con:

```
.em macro
```

11 Cadenas de texto

Al igual que se pueden manejar registros, groff permite manejar cadenas de texto. Las cadenas de texto se van a definir de una forma muy similar a cómo se definen las macros e incluso se le pueden pasar argumentos. Por ejemplo:

```
\" Con ds se define la cadena de texto
.ds ejemplo Esto es un ejemplo
\" Con \* se muestran los contenidos de la cadena de texto
\*[ejemplo]
```

Al ejecutar *[ejemplo] se mostrará la cadena de texto almacenada en "ejemplo". Como se ha dicho,

se les pueden pasar argumentos:

```
.ds ejemplo_hola Saludos \\$1 encantado de conocerte.

\*[ejemplo_hola Pepe]
.ejemplo_hola Juan
```

Este ejemplo devuelve la salida:

Saludos Pepe encantado de conocerte. Saludos Juan encantado de conocerte.

Incluso se pueden insertar comandos dentro de la cadena de texto:

```
.ds dingbats \f[ZDR]\N[\\$1]\f[]
\*[dingbats 50]\*[dingbats 51]
```

Este ejemplo devuelve:
🔪🔪

Para eliminar una cadena de texto se puede usar el comando .rm, por ejempo:

```
.ds dingbats \f[ZDR]\N[\\$1]\f[]
\*[dingbats 50]\*[dingbats 51]
.rm dingbats    \" Se elimina la cadena dingbats
\*[dingbats 50]\*[dingbats 51]    \" No devuelve nada
```

En el caso de introducir una cadena de texto que ocupe varías líneas, se usará el símbolo \

```
.ds ejemplo Este es\
un ejemplo\
que ocupa\
varias líneas.
\*[ejemplo]
.rm ejemplo
```

Esto da como resultado: Este esun ejemploque ocupavarias líneas.

Las cadenas de texto, macros y redirecciones (diversions) se pueden ejecutar de las mismas formas, es decir las podemos evaluar usando * o el . al comienzo de una línea.

```
.de simbolo
\f[ZDR]\N[\\$1]\f[]
..
Inicio
.simbolo 50
Hola\*[simbolo 50]ejemplo
.rm simbolo
```

Esto devuelve: Inicio 🔪 Hola🔪 ejemplo

Hay que fijarse en la falta de espacios en blanco a la ejecutarlo de una forma u otra.

Otras operaciones que se pueden hacer con las cadenas de texto son:

`.as nombre Cadena` Añade a la cadena ya existente "nombre" la cadena "Cadena".
`.length registro nombre` Almacena la longitud de la cadena "nombre" en el registro "registro".

12 Redirecciones (Diversions)

Lo habitual al trabajar con groff es redirigir el texto a la página en la que se está escribiendo. Pero hay situaciones en las que se necesita saber el tamaño del texto antes de enviarlo a la página. Por ejemplo, se necesita recuadrar un texto, habrá que saber lo que ocupa el texto antes de dibujar el recuadro. Para estas situaciones, se puede redirigir el texto a la memoria y una vez allí hacer las operaciones que debamos con él para finalmente redirigirlo a la página. A todo este proceso se le denomina redirección (diversion en inglés).

A las redirecciones se las van a tratar como a las macros. Para iniciar una redirección se usará:

```
.di identificador   \" Se inicia la redirección
.di                 \" Se termina la redirección
```

En la redirección se almacenarán líneas de texto completas, por lo que tendremos que forzar la ruptura de la línea con .br, en algunos casos, para que la línea se almacene completamente.

También se va almacenar la parte de la línea que se haya escrito ya en la página antes de ejecutar la redirección. Este comportamiento es útil cuando se usan las trampas de posición para trabajar con líneas enteras.

Un ejemplo:

```
Este es el comienzo de la línea
.di redireccion
se comienza la redirección.
.br
Se fuerza el salto de línea para almacenar la línea entera.
.br
.di
.redireccion
```

Cunado se ejecuta .redireccion nos da como resultado:

Este es el comienzo de la línea se comienza la redirección. Se fuerza el salto de línea para almacenar la línea entera.

Como vemos se ha almacenado la parte de la línea que estaba antes de ejecutar la redirección.

Vamos a quitar un salto de línea del ejemplo anterior:

```
Este es el comienzo de la línea
.di redireccion
```

```
se comienza la redirección.
.br
NO se fuerza el salto de línea para almacenar la línea entera.
.di
.redireccion
```

Ahora hay que fijarse en la extraña salida que se produce:

NO se fuerza el salto de línea para almacenar la línea entera. Este es el comienzo de la línea se comienza la redirección.

A veces son útiles las redirecciones .box que se diferencian a .di en que no se almacena la parte de la línea que ya se hubiese escrito:

```
Este es el comienzo de la línea
.box redireccion
se comienza la redirección.
.br
Se fuerza el salto de línea para almacenar la línea entera.
.br
.box

La salida es:
.br
.redireccion
```

Como se puede ver en la salida, el comienzo de la línea no está incluido en la redirección:

Este es el comienzo de la línea

La salida es:
se comienza la redirección. Se fuerza el salto de línea para almacenar la línea entera.

Después de terminar la redirección, los registros `\n[dn]` y `\n[dl]` almacenan el tamaño vertical y horizontal del texto almacenado en la redirección. Viene expresado en las unidades internas de groff. Por ejemplo:

```
.box redireccion
Texto a rodear.
.br
.box

.mk
.redireccion

.rt
\D'e \n[dl]u \n[dn]u'
```

Dibuja la salida:

En el caso de que la redirección cuente con otra línea encontraremos resultados inesperados:

```
.box redireccion
Texto a rodear.
.br
Más texto a rodear.
.br
.box

.mk
.redireccion

.rt
\D'e \n[dl]u \n[dn]u'
.rm redireccion
```

Como se puede comprobar en la salida, el tamaño vertical corresponde a 2 líneas y el horizontal a la línea más larga. Pero la redirección se limita a colocar las dos líneas de texto en la misma línea:

En el caso de tener que añadir más texto a una redirección que se tenga funcionando, se puede usar `.da` y `.boxa`.

12.1 Trampas en redirecciones (Diversion Traps)

Al igual que se puede crear trampas para el texto, se pueden crear trampas para las redirecciones. En este caso las trampas son exactamente igual que las trampas que se establecen con el comando .wh, pero dentro de una redirección se harán con .dt. La sintaxis será:

```
.dt distancia macro
```

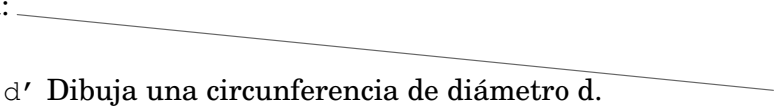
Donde la distancia es la distancia a la localización de la trampa y macro es la macro que se ejecutará al llegar a ella. Evidentemente, este comando sólo se puede ejecutar dentro de una redirección.

Si el comando .dt se ejecuta sin argumentos, se procede a eliminar la trampa.

13 Dibujando

Es posible realizar dibujos sin necesidad de insertar imágenes en groff. Para ello, se usará el comando \D. Se pondrán entre comillas el tipo de elemento de dibujo que se quiere realizar:

✎ `\D'l dx dy'` Dibuja una línea recta desde la posición actual a la posición relativa de coordenadas (dx,dy). Por ejemplo:

```
\D'l 10c 1c'
```
Dibujará: 

✎ `\D'c d'` Dibuja una circunferencia de diámetro d.

✎ `\D'C d'` Dibuja una circunferencia rellena de diámetro d.

✎ `\D'e dx dy'` Dibuja una elipse de diámetro horizontal dx y de diámetro vertical dy.

✎ `\D'E dx dy'` Dibuja una elipse rellena de diámetro horizontal dx y de diámetro vertical dy.

✎ `\D'a dx1 dy1 dx2 dy2'` Dibuja un arco dibujado en el sentido de las agujas del reloj que pasa por la posición actual, por (dx1,dy1) y finalmente por (dx2,dy2). La posición de (dx1,dy1) es relativa a la posición inicial. La posición de (dx2,dy2) es relativa a la de (dx1,dy1).

✎ `\D'~ dx1 dy1 dx2 dy2 ...'` Dibuja un spline que pasa por la posición actual respecto de la posición relativa de (dx1,dy1) y por (dx2,dy2),...

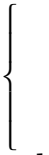
✎ `\D'p dx1 dy1 dx2 dy2 ...'` Dibuja un polígono que pasa por la posición actual, respecto de la posición relativa de (dx1,dy1) y por (dx2,dy2),... Por ejemplo:
```
\D'p 1c 0c 0c 1c'
```
Como se puede comprobar la posición de (dx1,dy1) es relativa a la posición inicial, la posición de (dx2,dy2) es relativa a (dx1,dy1), la posición de (dx3,dy3) es relativa a (dx2,dy2),...

✎ `\D'P dx1 dy1 dx2 dy2 ...'` Idéntico al apartado anterior, pero con la figura rellena.

✎ `\D't n'` Selecciona la anchura de la línea. Un valor de 0 selecciona la anchura de línea más fina que sea posible. Un valor negativo selecciona una anchura de línea proporcional al tamaño de punto actual.

✎ `\D'Fesquema componentes'` Cambia el color según el esquema que se le indique. Los esquemas son r (rgb), c (cmy), k (cmyk), g (escala de grises), d (color por defecto). Por ejemplo:
```
\D'Fg .5' \" Gris al 50%
\D'Fr #0000ff' \" Azul
```

✎ `\b'cadena de texto'` Apila una cadena de texto verticalmente y centradas. Se usa para construir paréntesis grandes o corchetes. Por ejemplo:

```
\b'\[lt]\[bv]\[lk]\[bv]\[lb]'
```


Estos elementos de dibujo, combinados con los comandos \v y \h (que sirven para cambiar la posición), se pueden usar para insertar símbolos que después e vayan a usar numerosas veces. Por ejemplo, el símbolo de Creative Commons:
```
\D't 0.3n'\v'-0.5m'\D'c 1.5m'\h'-1.32m'\v'0.26m'\f[HB]cc\f[]
```
Da como resultado:

14 Condicionales

En groff se pueden usar condicionales de forma que si la condición que se imponga es verdadera, se ejecutará el código que se le indique. Por ejemplo, si el registro x es igual a 1, se escriba un determinado texto:

```
.nr x 1
.if (\n[x] == 1) Verdadero \" Como x vale 1 escribe Verdadero
.nr x 2
.if (\n[x] == 1) Verdadero \" Como x vale 2 no escribe nada
```

Escribirá "Verdadero" una vez. La sintaxis de .if es:

```
.if expresión resultado
```

Si la expresión se evalúa verdadera, se ejecuta el resultado. En el caso de tener que escribir más de una línea, para el resultado, se deberá usar \{ y \}. Por ejemplo:

```
.nr x 1
.if (\n[x] > 0) \{
x es mayor de 0
y vale \n[x]
\}
.if (\n[x] < 0) \{
x es menor de 0
y vale \n[x]
\}
.if (\n[x] >= 0) \{
x es mayor o igual a 0
y vale \n[x]
\}
.if (\n[x] <= 0) \{
x es menor o igual a 0
y vale \n[x]
\}
.if (\n[x] == 1) \{
x es igual a 1
\}
.if !(\n[x] == 0) \{
x es distinto de 0
y vale \n[x]
\}
```

En el ejemplo se pueden ver los posibles comparadores que se pueden usar con los registros, == (igual a), >, <, >= (mayor o igual), <= (menor o igual). Se puede usar el operador ! para negar. En el ejemplo anterior:

```
.if !(\n[x] == 0) \{
x es distinto de 0
y vale \n[x]
\}
```

Se usa la comparación `!(\n[x] == 0)`, que tiene una ! al comienzo que hace que signifique "lo contrario de". En este caso significará que x no sea igual a 0. No hay que dejar espacio en blanco entre

la ! y el operador.

Se pueden usar operadores lógicos como el & (y). Por ejemplo:

```
.nr x 1
.if ((\n[x] > 0) & (\n[x] < 2)) Vale 1
```

La condición se cumplirá cuando sea mayor que cero y menor que 2. Es importante fijarse en el uso de los paréntesis.

Otro operador lógico es el : (o lógico). Por ejemplo:

```
.nr x 3
.if ((\n[x] < 0) : (\n[x] > 2)) No vale 1
```

En este caso la condición se cumple cuando x es menor de 0 o mayor a 2.

A veces hay que ejecutar una acción si la condición es cierta y otra diferente si la condición es falsa. Para ello, los lenguajes de programación definen la estructura if-else. En groff está estructura se logra con los comandos .ie y .el. Por ejemplo:

```
.nr x 3
.ie (\n[x] > 0) \{
Vale más de cero
\}
.el \{
Vale menos
\}
```

15 Bucles

Se han visto los condicionales, también es posible usar bucles para repetir ciertas tareas. Esto en groff se consigue con .while:

```
.nr x 10
.while (\n[x] > 0) \{
\n[x]
.nr x -1
\}
```

Este ejemplo muestra los números del 10 al 1. La sintaxis de .while es:

```
.while condición acción
```

Donde la condición se construye igual que para el condicional visto en el apartado anterior.

16 Escribiendo en disco

A veces es necesario escribir en el disco duro. Por ejemplo, en ocasiones es necesario poner un índice al texto. Lo normal será hacer una primera pasada con groff y cada vez que se vea un título guardar el número de página. Así se tendrá un archivo con el índice generado. En una segunda pasado, se puede coger el archivo con el índice e insertarlo en la posición en la que se quiera localizar el índice. En general, en todos los procesadores de texto son necesarias varias pasadas para obtener el índice correcto.

Para poder escribir en el disco, se deberá usar el modo "inseguro". Por defecto groff se ejecuta en modo "seguro" y no se permite la escritura en disco por parte de los comandos de groff.

Para poder ejecutar comandos en el modo "inseguro" se deberá usar la opción "-U":

```
groff -U -dpaper=a4 > manual_groff.ps
```

El ejemplo más sencillo de operación de escritura es escribir en un archivo. Esta acción la podemos realizar con los comandos .open y .opena. La diferencia entre ellos es que .open borra el archivo si existe previamente y .opena añade los contenidos al final del archivo en el caso de que el archivo ya exista.

La sintaxis de estos comandos es la siguiente:

```
.open identificador archivo
.opena identificador archivo
```

Por ejemplo, se va a abrir un archivo para escribir un mensaje:

```
.open Salida archivo.txt
.write Salida Hola mundo
.close Salida
```

También se puede usar el comando .writec para escribir en el fichero. La diferencia entre .write y .writec es que .write añade un salto de línea al final de cada texto que escribe.

Con el comando .so se puede incluir un archivo como ya se ha visto anteriormente. Deberemos usar el comando "soelim" para que estos archivos se incluyan, quedando los comandos que generan el archivo pdf de la siguiente forma:

```
soelim archivo.txt | preconv | eqn | tbl | groff-dpaper=a4 | ps2pdf - archivo.pdf
```

17 Generando un índice

Hasta ahora para generar los documentos, sólo se estaba haciendo una pasada con groff. Existen situaciones en las que hay que hacer varias pasadas hasta que el documento se obtiene correctamente. Este es, por ejemplo, el caso de los índices. Para generar un índice y colocarlo al inicio del documento, se tendrán que dar varias pasadas. Una para almacenar en un archivo la página de los títulos del documento. Si el índice se inserta al inicio, habrá que dar otra pasada para ver cómo varían las páginas, es de suponer que el índice puede llegar a ocupar varias páginas. Finalmente una tercera pasado para insertar el índice con los números de página correctos.

Hay una serie de registros que pueden ser útiles en la tarea de generar un índice:

Archivo actual: \n[.F]
Página actual: \n[%]

Ahora se puede generar una macro que vaya almacenado el número de página de cada título o de ca-

da referencia que se vaya a insertar.

Deberemos incluir el fichero generado en el documento con el comando ".so".

Por ejemplo:

```
...
\" Al inicio del documento se crea el archivo con el índice
.open archivoIndice indice.txt
.write archivoIndice Índice
.close archivoIndice
...
\" Se inserta el índice en la posición que se desee
.so indice.txt
...
\" En cada título o capítulo
Capítulo 3 Lo que sea
.opena archivoIndice indice.txt
.write archivoIndice Capítulo3 Lo que sea. página \n[%]
.close archivoIndice
...
```

Evidentemente esto queda mejor si se crean macros que hagan más cómodo el trabajo. Para generar el documento habrá que añadir la opción "-U" al comando, tal como se ha visto en el apartado anterior.

18 Tabulaciones

Muchas veces en un documento es necesario insertar tabuladores para colocar los datos o hacer el documento más legible. Para insertar un tabulador se debe usar \t. Para indicar la posición de los tabuladores con .ta y la distancia a la que debe estar el tabulador. Por ejemplo, para colocar los tabuladores a 2, 4, 6 y 8 centímetros:

```
.ds ejemplo  1 \t 2 \t 3 \t 4
.ta 2c 4c 6c 8c
\*[ejemplo]
```

Que generará la salida:

```
1       2       3       4
```

Si nos fijamos, sólo podremos usar los tabuladores como cadenas de texto. Podemos superar este inconveniente usando el siguiente truco:

```
.ds tabulador \t
.ta 2c 4c 6c 8c
1 \*[tabulador] 2 \*[tabulador] 3 \*[tabulador] 4
```

Que genera:

```
1        2        3        4
```

Titulo2 "Caracteres de relleno"

A veces interesa poner un carácter de relleno en lugar de espacios en blanco. Para ello se usa el comando .lc para indicar el carácter a usar. Se usará \a para indicar a Groff que debe usar el carácter de relleno. Por ejemplo:

```
.ds ejemplo 1\a2\t3\a4
.lc .
.ta 1c 5c 10c 15c
\*[ejemplo]
```

Que generará:

```
1.......2                    3.........................................4
```

18.1 Campos

Una forma más conveniente de organizar columnas de datos (a parte de las tablas) son los campos. Con los campos vamos a indicar primero la separación entre columnas. Después elegiremos un carácter que va a separar las columnas y otro carácter que indicará cuando debe usar ese espaciador.

```
\" Con .fc se le indican los caracteres de inicio y separador.
\" En este caso # inicia y - separa las columnas
.fc # -
\" Se le indica que las columnas estás separadas entre sí 5cm.
.ta T 5c
#1-2-3#
.br
#4-5-6#
.br
\" Si se usa .fc sin argumentos los caracteres delimitadores no funcionan.
.fc
#4-5-6#
```

Esto genera la salida:

```
1         2         3
4         5         6
#4-5-6#
```

19 Tablas

Las tabulaciones son útiles para casos sencillos, pero a veces la información queda más clara hacien-

do una tabla. Para poder usar las tablas tendremos que incluir el comando tbl en la cadena que genera el documento groff:

```
soelim archivo.txt | preconv | eqn | tbl | groff-dpaper=a4 | ps2pdf - archivo.pdf
```

Para generar las tablas se indicará con .TS y .TE el comienzo y el fin de una tabla. Para indicar el carácter que se usará para separar las columnas se usa "tab". Por ejemplo, para indicar que el carácter # será el separador de las columnas:

```
tab (#);
```

Se usará "center", "left" y "right" para conseguir que la tabla esté centrada, alineada a la izquierda o a la derecha.

Con "box" y "allbox" se consigue que la tabla esté rodeada por un recuadro o que todos los campos de la tabla tengan recuadro.

Hasta ahora para indicar una tabla se podría usar:

```
.TS
center box tab(#);
...
.TE
```

Vamos ahora a dar contenido a la tabla. Cada celda de la tabla puede estar centrada, alineada a la izquierda o a la derecha. Esto se indicará usando las letras "c", "l" y "r" respectivamente. Se usará el carácter "|" para dibujar una separación entre celdas.

Para decir a tbl que hemos acabado de especificar el formato y que debe empezar a admitir las columnas, se pondrá un punto.

Para indicarle a tbl que debe dibujar las líneas entre la filas se usará el carácter "_".

Todo junto en un ejemplo quedaría:

```
.TS
center box tab(#);
l | l l .
1#2#3
_
4#5#6
.TE
```

Esto genera la tabla:

1	2	3
4	5	6

Para que quede más claro se va a comentar línea por línea:

```
.TS
\" Se indica que la tabla va a estar centrada, rodeada por un marco y el
```

```
carácter de separación es "#":
center box tab(#);
\" La tabla consta de 3 columnas. La primera y la segunda están separadas
por una línea. El punto indica el final del formato:
l | l l .
\" Se comienzan a introducir las celdas:
1#2#3
\" Se dibuja una línea horizontal para separar filas:
_
4#5#6
.TE
```

En el caso de querer tener el formato de las celdas de una forma diferente, se pueden introducir varias líneas de formato:

```
.TS
center box tab(#);
l | l l
r r | c .
1#2#3
_
4#5#6
.TE
```

Se obtiene la salida:

1	2	3
4	5	6

Si se desea que una celda se extienda y ocupe otras, se usará el carácter "s" sobre las celdas que vaya a ocupar. Por ejemplo:

```
.TS
center box tab(#);
l s s
r | c c .
Ejemplo de celda que se extiende sobre otras
_
1#2#3
_
4#5#6
.TE
```

Como se puede ver en el resultado, la primera celda se extiende sobre las otras dos ocupando toda la cabecera:

Ejemplo de celda que se extiende sobre otras		
1	2	3
4	5	6

A veces hay que indicar que se desea que una celda tenga una determinada anchura, esto se hace con "w(tamaño)". Por ejemplo, para indicar que la celda debe medir 5cm de ancho "w(5c)":

```
.TS
center box tab(#);
l | l | lw(5c) .
1#2#3
_
4#5#6
.TE
```

En la salida se puede ver que la tercera columna tiene como ancho 5cm:

1	2	3
4	5	6

Puede interesar que una celda o columna tenga un determinado formato. Por ejemplo, que una celda esté en negrita. Para esta acción se usará "fB" para negritas y "fI" para cursiva:

```
.TS
center box tab(#);
l | lfB | lfI .
1#2#3
_
4#5#6
.TE
```

En la salida se puede observar que la segunda columna está en negrita y la tercera en cursivas:

1	**2**	*3*
4	**5**	*6*

Cuando el texto de una celda ocupa varias líneas se le indicará a tbl con:

```
T{
Texto largo
T}
```

Cuando se usa T{ ... T} Hay que fijarse que van en líneas distintas:

```
T{
Este es el texto
muy largo
T}
```

Un ejemplo:

```
.TS
center box tab(#);
l | lw(3c) | lw(1c) .
1#2#3
_
4#T{
Este es un texto que no cabe en una sola línea para el tamaño de celda que
se ha definido
T}#5
.TE
```

El texto de la celda central está forzado a ocupar una celda muy pequeña, por lo tanto debe ocupar varias líneas:

1	2	3
4	Este es un texto que no cabe en una sola línea para el tamaño de celda que se ha definido	5

Para saber más opciones a la hora de crear tablas, se puede consultar el manual de tbl escribiendo en un terminal:

```
man tbl
```

20 Ecuaciones matemáticas

Groff permite introducir ecuaciones matemáticas usando el programa "eqn". Para que funcione se deberá introducir en la cadena de generación del documento:

```
soelim archivo.txt | preconv | eqn | tbl | groff-dpaper=a4 | ps2pdf - ar-
chivo.pdf
```

Para introducir una ecuación usarán los comandos ".EQ" y ".EN":

```
.EQ
1 over 2
.EN
```

Que genera la fracción:

$$\frac{1}{2}$$

Para tener una ecuación en línea con el texto, sólo hay que pegar el texto a la fórmula $\frac{1}{2}$ de forma que Groff lo interpreta como una palabra más:

```
Para tener una ecuación en línea, sólo hay que pegar el texto a la fórmula
```

```
.EQ
1 over 2
.EN
```
de forma que Groff lo interpreta como una palabra más.

Importante: Según el paquete de macros que se tenga cargado, puede ser que no seamos capaces de hacer lo anterior, de hecho con algunos paquetes de macros se puede centrar, alinear a la izquierda o a la derecha simplemente añadiendo una letra:

```
\" Centrado:
.EQ C
...
.EN
\" Alineado a la izquierda
.EQ L
...
.EN
\" Indentado:
.EQ I
...
.EN
```

Además "eqn" tiene un mecanismo de delimitadores de forma que cuando, por ejemplo se escriba:

```
$x sup 2$
```
la ecuación se interprete. Para activar los delimitadores, sólo hay que usar "delim" con el carácter que se vaya a usar como delimitador. En el ejemplo anterior para usar el símbolo $ como delimitador:

```
.EQ
delim $$
.EN
```

Para desactivar los delimitadores:

```
.EQ
delim off
.EN
```

Hay algunos comandos interesantes a la hora de usar "eqn":

➠ `{} over {}` Sirve para escribir fracciones:

`.EQ`
`{x+2} over {x-1}`
`.EN`
	$\frac{x+2}{x-1}$

➠ `{} sup {}` Superíndice:

`.EQ`
`{(x+2)} sup {x-1}`
`.EN`
	$(x+2)^{x-1}$

También se puede usar para poner el índice en los radicales:

```
.EQ
"" sup 3 sqrt {x-1}
.EN
```

$\sqrt[3]{x-1}$

➠ `{} sub {}` Subíndice:

```
.EQ
{(x+2)} sub {x-1}
.EN
```

$(x+2)_{x-1}$

➠ `{} sqrt {}` Raíz cuadrada:

```
.EQ
sqrt {x-1}
.EN
```

$\sqrt{x-1}$

➠ `~` Espacio en blanco, "eqn" tiende a unir todas las palabras aunque estén separadas:

```
.EQ
a ~ b
.EN
```

$a\ b$

➠ `left ( right )` Paréntesis grandes. Se pueden usar también para {, }, [y]:

```
.EQ
left ( a over b right }
.EN
```

$\left(\frac{a}{b}\right\}$

➠ `{} from {} to {} {}` Índices para operadores grandes:

```
.EQ
sum from { k = 1 } to
{N+1} { k sup 2 }
.EN
```

$\sum_{k=1}^{N+1} k^2$

➠ `{} from {}` Índice inferior para operadores grandes:

```
.EQ
lim from {x -> inf} { x
sup 2 }
.EN
```

$\lim_{x\to\infty} x^2$

➠ `{} to {}` Índice superior para operadores grandes:

```
.EQ
sum to {x -> inf} { x
sup 2 }
.EN
```

$\sum^{\infty} x^2$

➠ `bold {}` Negritas:

`.EQ`
`bold {x} y`
`.EN`
	$\mathbf{x}y$

➠ `roman {}` Texto normal:

`.EQ`
`roman {x} y`
`.EN`
	$\mathrm{x}y$

➠ `"texto"` El texto entrecomillado no se trata como ecuación:

`.EQ`
`"Este texto no se trata como ecuación: a sup 2"`
`.EN`
	Este texto no se trata como ecuación: a sup 2

➠ `size tamaño {}` Cambia el tamaño de la letra:

`.EQ`
`size 14 {x} y`
`.EN`
	$x\,y$

➠ `matrix {`
`ccol { elemeto above elemento above ... }`
`ccol { elemeto above elemento above ... }`
`...`
`}`
Sirve para introducir una matriz. Con "ccol" se introducen las columnas. Dentro de cada columna "above" permite introducir el siguiente elemento:

`matrix {`
`ccol {1 above 2 above 3}`
`ccol {4 above 5 above 6}`
`}`
	$\begin{matrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{matrix}$

Puede que se necesiten introducir paréntesis o determinantes:

`matrix {`
`ccol {1 above 2 above 3}`
`ccol {4 above 5 above 6}`
`ccol {7 above 8 above 9}`
`}`
	$\begin{vmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{vmatrix}$

"ccol" ajusta los elementos al centro. "lcol" y "rcol" hacen lo mismo con ajustes a la izquierda y a la derecha.

➠ `lpile { elemeto above elemento above ... }`

Sirve para colocar elementos unos encima de otros, es una matriz con sólo una columna. Dentro de la columna "above" permite introducir el siguiente elemento:

```
signo(x) = left { lpile
{1 above 0 above -1}
~
lpile {si ~ x>0 above
si ~ x=0 above si ~ x<0
}
```

$$signo(x) = \begin{cases} 1 & si\ x > 0 \\ 0 & si\ x = 0 \\ -1 & si\ x < 0 \end{cases}$$

"lpile" ajusta los elementos a la izquierda. "cpile" y "rpile" hacen lo mismo con ajustes centrados y a la derecha.

Otros símbolos que pueden ser necesarios son:

+-	$\pm$
times	$\times$
sum	$\sum$
int	$\int$
left floor x right floor	$\lfloor x \rfloor$
left ceiling x right ceiling	$\lceil x \rceil$
prod	$\prod$
union	$\cup$
inter	$\cap$
>=	$\geq$
<=	$\leq$
==	$\equiv$
!=	$\neq$
->	$\rightarrow$
<-	$\leftarrow$
<<	$\ll$
>>	$\gg$
inf	∞
partial	∂
prime	$'$
approx	$\approx$
nothing	
cdot	$\cdot$
x dot	$\dot{x}$
x dotdot	$\ddot{x}$
x hat	$\hat{x}$
x tilde	$\tilde{x}$
x vec	$\vec{x}$
x dyad	$\overleftrightarrow{x}$
{xy} bar	$\overline{xy}$
{xy} under	$\underline{xy}$
grad	∇
DELTA	Δ
GAMMA	Γ

LAMBDA	Λ
OMEGA	Ω
PHI	Φ
PI	Π
PSI	Ψ
SIGMA	Σ
THETA	Θ
UPSILON	Υ
XI	Ξ
alpha	α
beta	β
chi	χ
delta	δ
epsilon	ε
eta	η
gamma	γ
iota	ι
kappa	κ
lambda	λ
mu	μ
nu	ν
omega	ω
omicron	o
phi	ϕ
pi	π
psi	ψ
rho	ρ
sigma	σ
tau	τ
theta	θ
upsilon	υ
xi	ξ
zeta	ζ

20.1 Extendiendo eqn con definiciones

"eqn" permite hacer definiciones que después se podrán usar en las fórmulas. Por ejemplo, se quiere introducir un texto muy largo y se desea escribir poco:

```
define nombre 'texto'
```

Cada vez que se use el nombre dado en la definición, se sustituirá por el texto:

```
.EQ
define xx 'sqrt {1 over 2}'
f(x) = xx over 2
.EN
```

$$f(x) = \frac{\sqrt{\frac{1}{2}}}{2}$$

Una vez que se hace una definición en una fórmula, se puede usar en el resto de las fórmulas:

```
.EQ
define xx 'sqrt {1 over 2}'
.EN
...
En otro punto del documento la definición de xx es reconocida:
.EQ
f(x) = xx over 2
.EN
```

Usando "special" se pueden usar macros de groff dentro de las fórmulas. Para ello cuando la macro es llamada, la cadena "0s" (cero ese) debe contener el texto que se escribirá en la fórmula. También se deberán dar modificar, si es necesario, a los siguientes registros:
`.0w` La anchura de la salida
`.0h` La altura de la salida
`.0d` La profundidad de la salida
`.0skern` Subíndice kern. Indica a cuánto debe colocarse un subíndice en ese objeto.
`.0skew` La inclinación. Indica hasta que punto a la derecha del centro del objeto debe colocarse un acento sobre el objeto.

Por ejemplo, esta macro sólo inserta el símbolo ⇒ en la ecuación:

```
.EQ                                                    ⇒
define flecha 'special FlechaMacro'
.EN
.de FlechaMacro
.ds 0s \\f[S]\\N[222]\\f[]
..
.EQ
flecha x
.EN
```

Vemos que para usar la definición le tenemos que pasar un argumento. Aunque en este caso la definición no usa el argumento:

```
.EQ
flecha x
.EN
```

En el siguiente ejemplo, se usa el argumento, al cual se accede con "*[0s]":

```
.EQ
define entreFlechas 'special EntreFlechasMacro'
.EN
.de EntreFlechasMacro
.ds 0s \\f[S]\\N[222]\\f[] \*[0s] \\f[S]\\N[220]\\f[]
..
.EQ
entreFlechas x
.EN
```

$\Rightarrow x \Leftarrow$

Es importante darse cuenta de que eqn pasa a través de 0s, 0w, 0h, 0d, 0skern y 0skew, el argumento (0s) y el resto de las propiedades del argumento.

Un ejemplo más avanzado, tachar el argumento:

```
.EQ
define cancel 'special Ca'
.EN
.de Ca
.  ds 0s \
\Z'\\*[0s]'\
\v'\\n[0d]u'\
\D'l \\n[0w]u -\\n[0h]u-\\n[0d]u'\
\v'\\n[0h]u'
..
```

Cada vez que se escriba "cancel {argumento}", el argumento se tachará:

```
.EQ
cancel {222}
.EN
```

~~222~~

Otro ejemplo, dibujar una caja alrededor del argumento:

```
.EQ
define box 'special Bx'
.EN
.de Bx
.  ds 0s \
\Z'\h'1n'\\*[0s]'\
\Z'\
\v'\\n[0d]u+1n'\
\D'l \n[0w]u+2n 0'\
\D'l 0 -\\n[0h]u-\\n[0d]u-2n'\
\D'l -\\n[0w]u-2n 0'\
\D'l 0 \\n[0h]u+\\n[0d]u+2n'\
```

```
'\
\h'\\n[0w]u+2n'
.  nr 0w +2n
.  nr 0d +1n
.  nr 0h +1n
..

.EQ
box {xyz} sup 2
.EN
```

21 Paquetes de macros

En este documento se ha creado un documento para groff desde cero, pero hemos tenido que crear a mano todos los elementos que se tuviesen que usar. Existen paquetes de macros ya creados que permiten acelerar mucho el trabajo. Lo más común será usar uno de estos paquetes de macros y modificarlos a nuestra conveniencia.

Entre los paquetes más conocidos están "ms" usados para escribir libros, informes, manuales,... Este paquete provee macros para cabeceras, notas al pie, listas, tablas de contenidos,... Para usar este paquete, se deberá añadir la opción "-ms" al comando groff.

El paquete más importante a la hora de usar groff es "man". El paquete de macros "man" es usado para crear las páginas de manual de los programas en GNU/Linux y que se pueden leer con el comando "man".

Más información sobre estos paquetes se pueden encontrar en el manual de referencia de groff.

Apéndice 1: Tabla de caracteres para Zapf Dingbats y Symbol

Tabla de caracteres del tipo de letra Zapf Dingbats:

32		60	✣	88	✸	116	▼	144		172	①	200	⑨	228	➤
33	✁	61	✝	89	✹	117	◆	145		173	②	201	⑩	229	➥
34	✂	62	✞	90	✺	118	❖	146		174	③	202	❶	230	➦
35	✃	63	✟	91	✻	119	◗	147		175	④	203	❷	231	➧
36	✄	64	✠	92	✼	120	❙	148		176	⑤	204	❸	232	➨
37	✆	65	✡	93	✽	121	❚	149		177	⑥	205	❹	233	➩
38	✇	66	☛	94	✾	122	❯	150		178	⑦	206	❺	234	➪
39	✈	67	☞	95	✿	123	❛	151		179	⑧	207	❻	235	➫
40	✉	68	✌	96	❀	124	❜	152		180	⑨	208	❼	236	➬
41	☛	69	✍	97	❁	125	❝	153		181	⑩	209	❽	237	➭
42	☞	70	✎	98	❂	126	❞	154		182	❶	210	❾	238	➮
43	✍	71	✏	99	❃	127		155		183	❷	211	❿	239	➯
44	✌	72	★	100	❄	128	❨	156		184	❸	212	➔	240	
45	✎	73	☆	101	❅	129	❩	157		185	❹	213	→	241	➱
46	✏	74	✪	102	❆	130	❪	158		186	❺	214	↔	242	➲
47	✐	75	✯	103	❇	131	❫	159		187	❻	215	↕	243	➳
48	✑	76	✰	104	❈	132	❬	160		188	❼	216	↘	244	➴
49	✒	77	✱	105	❉	133	❭	161	❡	189	❽	217	→	245	➵
50	✓	78	✲	106	❊	134	❮	162	❢	190	❾	218	↗	246	➶
51	✔	79	✳	107	❋	135	❯	163	❣	191	❿	219	→	247	➷
52	✕	80	✴	108	●	136	❰	164	❤	192	①	220	➔	248	➸
53	✖	81	✵	109	❍	137	❱	165	❥	193	②	221	→	249	➹
54	✗	82	✶	110	■	138	❲	166	❦	194	③	222	→	250	➺
55	✘	83	✷	111	❑	139	❳	167	❧	195	④	223	➟	251	➻
56	✙	84	✸	112	❒	140	❴	168	♣	196	⑤	224	➠	252	➼
57	✚	85	✹	113	❑	141	❵	169	♦	197	⑥	225	➡	253	➽
58	✛	86	✺	114	❒	142		170	♥	198	⑦	226	➢	254	➾
59	✜	87	✻	115	▲	143		171	♠	199	⑧	227	➢	255	

Tabla de caracteres del tipo de letra Symbol:

32		60	<	88	Ξ	116	τ	144		172	←	200	∪	228	™
33	!	61	=	89	Ψ	117	υ	145		173	↑	201	⊃	229	∑
34	∀	62	>	90	Z	118	ϖ	146		174	→	202	⊇	230	⎛
35	#	63	?	91	[	119	ω	147		175	↓	203	⊄	231	⎜
36	∃	64	≅	92	∴	120	ξ	148		176	°	204	⊂	232	⎝
37	%	65	A	93	]	121	ψ	149		177	±	205	⊆	233	⎡
38	&	66	B	94	⊥	122	ζ	150		178	″	206	∈	234	⎢
39	϶	67	X	95	_	123	{	151		179	≥	207	∉	235	⎣
40	(	68	Δ	96	‾	124	|	152		180	×	208	∠	236	⎧
41	)	69	E	97	α	125	}	153		181	∝	209	∇	237	⎨
42	*	70	Φ	98	β	126	~	154		182	∂	210	®	238	⎩
43	+	71	Γ	99	χ	127		155		183	•	211	©	239	⎪
44	,	72	H	100	δ	128		156		184	÷	212	™	240	
45	−	73	I	101	ε	129		157		185	≠	213	∏	241	⟩
46	.	74	ϑ	102	ϕ	130		158		186	≡	214	√	242	⌡
47	/	75	K	103	γ	131		159		187	≈	215	·	243	⌠
48	0	76	Λ	104	η	132		160	€	188	…	216	¬	244	⎮
49	1	77	M	105	ι	133		161	ϒ	189	|	217	∧	245	⌡
50	2	78	N	106	φ	134		162	′	190	—	218	∨	246	⎞
51	3	79	O	107	κ	135		163	≤	191	↵	219	⇔	247	⎟
52	4	80	Π	108	λ	136		164	⁄	192	ℵ	220	⇐	248	⎠
53	5	81	Θ	109	μ	137		165	∞	193	ℑ	221	⇑	249	⎤
54	6	82	P	110	ν	138		166	*f*	194	ℜ	222	⇒	250	⎥
55	7	83	Σ	111	ο	139		167	♣	195	℘	223	⇓	251	⎦
56	8	84	T	112	π	140		168	♦	196	⊗	224	◊	252	⎤
57	9	85	Y	113	θ	141		169	♥	197	⊕	225	⟨	253	⎭
58	:	86	ς	114	ρ	142		170	♠	198	∅	226	®	254	⎦
59	;	87	Ω	115	σ	143		171	↔	199	∩	227	©	255	

Tabla de caracteres estándar:

32		60	<	88	X	116	t	144	ı	172	¬	200	È	228	ä
33	!	61	=	89	Y	117	u	145		173	–	201	É	229	å
34	"	62	>	90	Z	118	v	146	`	174	®	202	Ê	230	æ
35	#	63	?	91	[	119	w	147	˝	175	¯	203	Ë	231	ç
36	$	64	@	92	\	120	x	148	˙	176	°	204	Ì	232	è
37	%	65	A	93	]	121	y	149	˘	177	±	205	Í	233	é
38	&	66	B	94	ˆ	122	z	150	ˇ	178	²	206	Î	234	ê
39	’	67	C	95	_	123	{	151	°	179	³	207	Ï	235	ë
40	(	68	D	96	‘	124			152	˛	180	´	208	Đ	236	ì
41	)	69	E	97	a	125	}	153	“	181	µ	209	Ñ	237	í
42	*	70	F	98	b	126	~	154	”	182	¶	210	Ò	238	î
43	+	71	G	99	c	127		155	œ	183	·	211	Ó	239	ï
44	,	72	H	100	d	128	,	156	ł	184	¸	212	Ô	240	ð
45	-	73	I	101	e	129	«	157	„	185	¹	213	Õ	241	ñ
46	.	74	J	102	f	130	»	158	Œ	186	º	214	Ö	242	ò
47	/	75	K	103	g	131	•	159	Ł	187	›	215	×	243	ó
48	0	76	L	104	h	132	*f*	160		188	¼	216	Ø	244	ô
49	1	77	M	105	i	133	⁄	161	¡	189	½	217	Ù	245	õ
50	2	78	N	106	j	134	‰	162	¢	190	¾	218	Ú	246	ö
51	3	79	O	107	k	135	†	163	£	191	¿	219	Û	247	÷
52	4	80	P	108	l	136	‡	164	¤	192	À	220	Ü	248	ø
53	5	81	Q	109	m	137	–	165	¥	193	Á	221	Ý	249	ù
54	6	82	R	110	n	138	—	166	¦	194	Â	222	Þ	250	ú
55	7	83	S	111	o	139		167	§	195	Ã	223	ß	251	û
56	8	84	T	112	p	140	fi	168	¨	196	Ä	224	à	252	ü
57	9	85	U	113	q	141	fl	169	©	197	Å	225	á	253	ý
58	:	86	V	114	r	142		170	ª	198	Æ	226	â	254	þ
59	;	87	W	115	s	143		171	‹	199	Ç	227	ã	255	ÿ