

## Búsqueda de archivos de texto con grep

# De Cacería en el Disco Duro

Los eruditos de la Edad Media hubiesen vendido sus almas a cambio de la ingente cantidad de literatura que abunda actualmente en Internet. En la actualidad, los documentos desordenan nuestros discos duros debido a las descargas indiscriminadas. ¿Cómo encontramos exactamente un pasaje de un texto en nuestra base de datos digital? El comando del shell *grep* nos puede ayudar a encontrar esa cita escurridiza. **POR ELISABETH BAUER**



Las cosas que no seamos capaces de recordar las debemos guardar en nuestro ordenador, reza una máxima del usuario informático.

Ésta no es una mala idea, pero, a diferencia de lo que ocurre con la memoria humana, que normalmente recuperará la información almacenada de forma fidedigna (excepto en los exámenes finales, por supuesto), no siempre es tan fácil encontrar información en nuestro disco duro. Podemos perder mucho tiempo en encontrar un archivo del que hemos olvidado su nombre o dónde fue almacenado. Incluso el saber exactamente que archivo contiene la información que buscamos puede ser de poca ayuda en caso de archivos de texto grandes.

El comando del shell *grep*, que localiza cadenas de texto en ficheros es útil en ambos casos. En la situación más simple, podemos ejecutar *grep* con la tecla de búsqueda y el archivo a buscar. *grep* nos mostrará todas las líneas en el archivo especificado que contengan el texto buscado. Imaginemos que deseamos buscar en el archivo *biblia.txt* el texto "Jardín del Edén". Debemos escribir

```
grep Edén biblia.txt
```

en la línea de comandos y *grep* nos mostrará los pasajes apropiados del archivo. Si la búsqueda contiene espacio debemos utilizar comillas. Por ejemplo:

```
grep "Jardín del Edén"
biblia.txt
```

Debemos prestar atención y esperar fallos cuando usemos caracteres especiales: \*, ? y ! tienen un significado especial para la línea de comandos. Otro grupo de caracteres (., \*<C>, ^<C>, \$ y \) no se interpreta por *grep* tal cual. En su lugar, la herramienta supondrá una expresión regular. El lado positivo es que esto nos permite construir búsquedas muy potentes y complejas, si bien es posible que prefiramos evitar estos caracteres con *grep* hasta que nos sintamos a gusto con la herramienta.

Si no sabemos que archivo contiene el texto que buscamos, podemos ejecutar *grep* con un comodín. La obra "Moby Dick" del autor Herman Melville está compuesta de una colección de archivos de texto.

```
grep blanco moby.*
```

nos mostrará todas las veces que la palabra *blanco* aparece en la obra maestra de Melville (ver figura 1). El asterisco significa "cualquier grupo de letras". El interfaz cambiará esta expresión por el nombre de cualquier archivo en el directorio actual que empiece por los caracteres *moby.*

Si los archivos en los que deseamos buscar están distribuidos por diversos directorios debemos añadir la opción *r* para indicarle a *grep* que busque en una carpeta completa:

```
grep -r blanco Melville/obras/
```

buscará *blanco* en el directorio *obras* y en los subdirectorios que cuelguen de él.

### VISIÓN GENERAL DE GREP

Comando	Acción
<i>grep patrón fichero</i>	búscas en un archivo un patrón
<i>grep patrón *.htm</i>	busca en todos los archivos en un directorio que acaban con el sufijo .htm
<i>grep -r patrón directorio</i>	realiza una búsqueda recursiva en un directorio y sus subdirectorios.
<i>grep -i patrón fichero</i>	ignora la diferencia entre mayúsculas y minúsculas.
<i>grep -A n</i>	muestra las siguientes n líneas tras la línea que contiene la búsqueda.

```

john@black.box: /home/john/Melville - Shell - Konsole
Session Edit View Settings Help
[john@black Melville]# wget http://www.ibiblio.org/gutenberg/etext91/moby.zip
--09:58:21--  http://www.ibiblio.org/gutenberg/etext91/moby.zip
       >
Resolving www.ibiblio.org... done.
Connecting to www.ibiblio.org[152.2.210.81]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 606,033 [application/zip]

100%[=====] 606,033      59,22KB/s  ETR 00:00

09:58:32 (59,22 KB/s) - 'moby.zip' saved [606033/606033]

[john@black Melville]# unzip -q moby.zip
[john@black Melville]# ls
moby.0  moby.11  moby.121  moby.134  moby.25  moby.37  moby.49  moby.60  moby.73  moby.85  moby.97
moby.1  moby.110  moby.123  moby.14  moby.26  moby.38  moby.5  moby.61  moby.74  moby.86  moby.98
moby.10  moby.111  moby.124  moby.15  moby.27  moby.39  moby.50  moby.62  moby.75  moby.87  moby.99
moby.100  moby.112  moby.125  moby.16  moby.28  moby.4  moby.51  moby.63  moby.76  moby.88  moby.zip
moby.101  moby.113  moby.126  moby.17  moby.29  moby.40  moby.52  moby.64  moby.77  moby.89  README
moby.102  moby.114  moby.127  moby.18  moby.3  moby.41  moby.53  moby.65  moby.78  moby.9  README
moby.103  moby.115  moby.128  moby.19  moby.30  moby.42  moby.54  moby.66  moby.79  moby.90
moby.104  moby.116  moby.129  moby.2  moby.31  moby.43  moby.55  moby.67  moby.8  moby.91
moby.105  moby.117  moby.13  moby.20  moby.32  moby.44  moby.56  moby.68  moby.80  moby.92
moby.106  moby.118  moby.130  moby.21  moby.33  moby.45  moby.57  moby.69  moby.81  moby.93
moby.107  moby.119  moby.131  moby.22  moby.34  moby.46  moby.58  moby.7  moby.82  moby.94
moby.108  moby.12  moby.132  moby.23  moby.35  moby.47  moby.59  moby.70  moby.83  moby.95
moby.109  moby.120  moby.133  moby.24  moby.36  moby.48  moby.6  moby.71  moby.84  moby.96
[john@black Melville]# grep white moby.*
moby.100:withdrawing it from the fold that had hidden it, he held up a white arm of
moby.100:a bouncing great whale, with a milky-white head and hump, all crows' feet and
moby.100:great-grandfather, with the white head and hump, runs all afoam into the pod.
moby.100:First, the white hump backed through the wreck, as though it was all chips.
moby.102:great, white, worshiped skeleton lay lounging --a gigantic idler! Yet, as the
moby.103:width, and looks something like a white billiard-ball. I was told that there
moby.108:the white heat for this kind of fine work. Um-m. So he must. I do deem it
moby.109:Mat-snai, and Sikoke. With his snow-white new ivory leg braced against the
moby.110:uncontaminated seas, interflow with the blue heavens; and so form the white
moby.113:the White Whale? For the white fiend! But now for the bars; thou must
moby.118:with three tapering white flames, each of the three tall masts was silently
moby.119:The parted mouth of Tashtego revealed his shark-white teeth, which strangely
moby.119:it's mark it well; the white flame but lights the way to the White Whale!
moby.125:white, for I will not let this go.
moby.126:white bubbles in the blue of the sea. The life-buoy --a long slender cask --was
moby.128:and while they were yet in swift chase to windward, the white hump and head of
moby.128:bubbling white water; and after that nothing more; whence it was concluded

```

Figura 1: El libro "Moby Dick" de Melville está subdividido en una serie de archivos de texto. El comodín "moby.\*" indica a *grep* que busque en todos los archivos del directorio.

## El Trio: ps, grep y kill.

*grep* no es sólo útil para búsquedas de texto filosóficos y teológicos, si no que puede ser combinado con otros comandos del shell. Si la respuesta de un comando produce mucho texto, *grep* puede ser utilizado escribiendo tras la instrucción el carácter "|" y *grep texto\_a\_buscar* para filtrar el resultado y mantener solo las partes en las que estamos interesados. Un caso típico es en el que usaríamos *grep* para cerrar un programa que se ha bloqueado.

El comando *ps ax* nos muestra los procesos activos. Podemos usar *grep* para aplicar un filtro y encontrar solo el programa que estamos buscando, Mozilla por ejemplo:

```
> ps ax | grep mozilla
2500 ? S 1:40 /usr/lib/B
mozilla-1.3/mozilla-bin
5645 pts/4 S 0:00 grep mozilla
```

*grep* nos muestra dos coincidencias que contienen Mozilla: el buscador y el propio *grep*. La parte que nos interesa para cerrar el programa aparece al principio de cada línea: el ID del proceso. Ahora podemos escribir *kill 2500* para cerrar el programa deseado.

Debido a que los gurús del shell suelen ser notablemente perezosos, necesitamos encontrar un método para no tener que escribir este comando cada vez que lo requiramos: en otras palabras, necesitamos un *alias*. Los alias definidos deben ser guardados en el archivo *.bashrc* en el directorio raíz. Este archivo se ejecuta cada vez que abrimos un shell de comandos interactivo. Usaremos nuestro editor de texto favorito para abrir el fichero, por ejemplo *kwite ~/.bashrc* & o *vi ~/.bashrc* en el escritorio, dependiendo de nuestras preferencias. Debido a que *vi* no es sencillo, veamos algunos comandos simples. Si escribimos *G* le indicamos a *vi* que debe ir al final del archivo. Entonces podemos escribir *o* para cambiar el editor al modo de introducción de datos. A diferencia de los comandos *a* y *i*, *o* indica a *vi* que inicie la inserción en la línea donde está situado el cursor.

Ahora podemos introducir nuestro alias en la última línea de *.bashrc*. En lugar de *pss* podemos usar cualquier otro nombre fácil de recordar pero evitando usar el nombre de un comando existente:

```
alias pss="ps ax |BB
grep"
```

Ahora presionamos *[Esc]* *ZZ* o *[Escape]:wq* para almacenar el archivo y salir de *vi*. Para utilizar nuestro nuevo alias en el shell, necesitamos analizar sintácticamente el archivo de configuración. Para hacerlo debemos escribir:

```
~/.bashrc
```

Ahora podemos usar el comando *pss nombreprograma* para buscar un programa activo.

## Libro de Direcciones

*grep* es tremendamente flexible. Una de mis aplicaciones favoritas para *grep* es un sencillo libro de direcciones.

Todo lo que necesitamos son los comandos *grep*, *alias* y *cat* y un fichero de texto donde podamos almacenar los nombres, números de teléfono y las direcciones de correo electrónico y de correo ordinario de nuestros amigos, conocidos y parientes. Una entrada puede ser como:

```
Charly Pingüino
+12345 678
tux@linux.org
C/del Polo Sur
Villatux, Antártica
```

Ahora debemos salvar el archivo como *direcciones* en nuestro directorio raíz y añadir el siguiente alias a *.bashrc*:

```
alias tel="cat ~/direcciones |BB
grep -i -A 4"
```

El comando *cat* nos muestra el contenido del archivo *direcciones*. El carácter | envía este resultado a *grep*. La opción *-i* garantiza que la búsqueda no diferenciará entre mayúsculas y minúsculas. Finalmente, *-A 4* indica a *grep* que debe mostrar las 4 líneas siguientes a la primera coincidencia con la búsqueda.

De nuevo

```
~/.bashrc
```

analizará sintácticamente de nuevo nuestro archivo de configuración. En el futuro, solo necesitaremos escribir *tel nombre* en el shell para recuperar la dirección de la persona que busquemos. ■