

Entendiendo lo básico de Bash

# ENTRENAMIENTO BASH

Andrey Tsidvinsev / 123 RF

Incluso los expertos olvidan lo fundamental. Aprendemos más – o refrescamos la memoria – de lo básico de Bash. **POR BRUCE BYFIELD**

Una desventaja del sobreuso del escritorio es que la gente aprende sobre la línea de comandos sólo cuando la necesitan. Como resultado, su conocimiento es a menudo desordenado y está lleno de lagunas. Por ejemplo, durante años he estado usando el comando *su* varias veces al día para registrarme como root desde un terminal virtual en el escritorio de mi cuenta diaria. Siempre pensé que cuando has acabado como root, no se puede volver a la cuenta de uso diaria en el mismo terminal; en su lugar, teníamos que cerrar la ventana y abrir otra. Entonces comprendí, por pura casualidad, que todo cuanto necesitaba hacer era escribir *exit*.

Desde entonces, he descubierto que, si se les presiona, incluso los expertos confesarán lagunas en su conocimiento.

## Vuelta a lo Básico

En este artículo discutiré un tema que debí haber tratado hace meses: lo más básico de Bash, la línea de comandos utilizada por la mayoría de las distribuciones GNU/Linux. Trataré las herramientas básicas y los recursos para información, además de

los básicos para navegación. Incluso conociendo algunos de estos datos, una discusión sistemática llenará las lagunas en el conocimiento. Si no, siempre podemos pasar la información a algún usuario de escritorio para su provecho.

La línea de comandos es una de las interfaces más básicas inventadas. Lo único que tenemos que hacer es introducir una cadena de caracteres para obtener los resultados deseados. ¿Si?

Bueno, no del todo.

Este método funciona si enfocamos la línea de comandos como una serie de hechizos, cada uno de los cuales debemos copiar cuidadosamente antes de pulsar la tecla Enter. Sin embargo, si añadimos nuestras propias entradas, incluso el nombre de un fichero guardado, con frecuencia necesitamos saber más.

## Sensibles a Mayúsculas/Minúsculas

Para empezar necesitamos saber que, a diferencia de la línea de comandos de Windows, Bash es sensible a mayúsculas y minúsculas, tratando estas letras de manera distinta. Por esta razón, los usua-

rios experimentados tienden a evitar usar mayúsculas para todo: no desean molestar con tecleos extra.

Lo que es más importante, esto también significa que un fichero llamado *taxsummary* no es el mismo que otro llamado *Taxsummary*. Si olvidamos este factor fundamental, podemos perder mucho tiempo intentando localizar un fichero.

## Elección de Carácter

Además, algunos caracteres no están permitidos en la línea de comandos con objeto de evitar confusión. Los caracteres prohibidos incluyen un espacio, el cual separa las partes de un comando; la barra inclinada hacia delante (*'/'*), que separa los nombres de los directorios en la ruta de ficheros (no una barra diagonal invertida, *'\'*, como en Windows); y caracteres tales como el interrogante o asterisco, los cuales se usan en expresiones regulares o como comodines.

Estas restricciones pueden causar algunos conflictos con el escritorio, que posee reglas más relajadas, especialmente en lo que se refiere al uso de espacios en los

nombres de ficheros. Sin embargo, podemos eludir estas restricciones en algunos casos. La manera más fácil es simplemente evitar usar nada más exótico que un punto en los nombres de ficheros y usar guiones simples o bajos en vez de espacios, tal y como se hacía en el viejo Unix. Podemos usar comillas simples o dobles en torno al nombre del fichero con espacios o caracteres restringidos en ellos como una señal de que los nombres no deberán ser leídos como se hace habitualmente, sino literalmente:

```
$ cat mi fichero # MAL
$ cat "mi fichero" # BIEN
```

O, si preferimos, podemos hacer lo que se llama “escapar” y situar una barra diagonal delante de un carácter. Dicha barra diagonal es señal de que el carácter siguiente – y sólo el siguiente – no será leído normalmente. Por ejemplo, si introducimos `mi\fichero`, le estaremos diciendo a Bash que lea el espacio entre las dos palabras como parte de un nombre de fichero continuo. Sin la barra diagonal, Bash podría leer nuestra entrada como dos ficheros, `mi` y `fichero`.

### Estructura del Comando

Un comando típico posee tres partes que aparecen en un orden establecido:

- **El comando en sí** – La acción fundamental que estamos llevando a cabo. A veces, introducir un comando hará algo básico. Por ejemplo, si escribimos `cal`, entonces la salida es un calendario para el mes en curso. A menudo, sin embargo, el comando por sí mismo no hace nada.
- **Opciones, argumentos, variables o parámetros** – Variaciones de lo que hace el comando. Las opciones que fueron desarrolladas por Unix o por desarrolladores sin afiliación son una sola letra con un guión delante de ella. Las opciones desarrolladas por el proyecto GNU son palabras completas con dos guiones delante de ellas. A menudo, un comando tendrá una de cada una que hace lo mismo y podremos escoger lo que deseemos – o la velocidad de una opción de guión único o la claridad de la opción de los dos guiones. Algunas opciones también acaban con una especificación; por ejemplo, si usamos `-tvfat` con el comando `mount`, estamos indicando

que deseamos añadir un dispositivo a nuestro sistema que está formateado para VFAT o FAT32, el viejo formato utilizado años ha por Windows. Las opciones varían entre los comandos, aunque encontraremos algunas utilizadas frecuentemente casi universalmente, tales como `-r` o `R`, que suele significar que el comando afecta recursivamente a los subdirectorios y a sus contenidos, y `-v`, que le dice al comando que muestre una explicación de lo que está haciendo.

- **Entrada** – El material de trabajo para un comando. Este puede ser un fichero único, como cuando estamos utilizando el comando `rm` para borrar un fichero. Otras veces, como con el comando `mv`, puede ser el nombre de un fichero original seguido por un espacio y la nueva ruta para el fichero. También puede ser una cadena de caracteres por la que estamos buscando o una especificación, como puede ser el formato empleado por el comando `date`.

Algunos comandos tienen también subcomandos que van directamente detrás de él, como ocurre con los subcomandos `install` o `remove` usados por `apt-get`, la herramienta utilizada para instalar paquetes en Debian y Ubuntu.

Probablemente haya notado, además, que muchos comandos emplean abreviaturas que describen su función. Por ejemplo, `cp` copia ficheros y `ls` los lista. Esta información es muy útil cuando tenemos alguna idea de qué deseamos hacer, pero no estamos seguros de qué comando necesitamos.

### Atajos de Teclado de la Línea de Comandos

Una razón por la que Bash es más fácil de usar que la línea de comandos de Windows, es que posee algunas herramientas integradas para facilitarnos las cosas. Estas herramientas pueden ser grandes

ahorradoras de tiempo – sin contar la ayuda que supone para nuestra memoria.

La primera de dichas herramientas es el historial de comandos, que guarda los comandos que introducimos. Podemos utilizar las teclas de flecha arriba y abajo para movernos a través del historial y ahorrarnos tener que reescribir comandos largos o difíciles de recordar.

El comando historial incluye todos los tipos de trucos interesantes, la mayoría de los cuales se abordarán otro día. Sin embargo, un filtro básico es hacer referencia a una entrada previa introduciendo `![número]` y pulsando `Ctrl + j`. Por ejemplo, si ejecutamos el comando `ls` hace tres entradas, escribiendo `!-3` nos presentaría automáticamente el comando `ls`, pero sin ejecutarlo. De manera alternativa, podríamos usar la estructura `![cadena]` para encontrar la última entrada en el historial que coincidió con la cadena.

Otro ahorrador de tiempo es el auto-completado con tabulador. Cuando hemos introducido parte de una ruta o un comando, podemos pulsar la tecla `Tab` para completarlo. Si existe más de una posibilidad, Bash nos las presentará en formato multi-columna, y podemos continuar introduciendo letras hasta que sólo quede una posibilidad.

Por ejemplo, si introducimos `ch`, y a continuación pulsamos la tecla `Tab`, veremos una lista con 22 posibles opciones. Continuamos introduciendo una `a`, y pulsamos `Tab` de nuevo. Las posibilidades se habrán reducido a 5. Añadimos `t` y pulsamos `Tab`, y esta vez, la única elección es `chattr`, cuyas letras finales se añaden automáticamente.

Si, como mucha gente, estamos usando un terminal virtual, también tenemos disponible la funcionalidad básica de copia y pegado. Podemos utilizar los comandos básicos listados en el menú `Edit` de nuestro terminal, aunque si preferimos, podemos resaltar algún texto, mover luego el cursor a la posición a la que deseamos

```
bruce@nanday:~$ ch
chacl      chcon      cheetah    chown
chage      cheatmake  cheetah-compile  chrt
charmap    checkisomd5  chfn       chsh
charset    checkmodule  chgrp      chvt
chattr     checkpolicy  chkdupexe
chcat      cheese      chmod
bruce@nanday:~$ cha
chacl      chage      charmap    charset    chattr
bruce@nanday:~$ chattr
```

Figura 1: El autocompletado con tabulación puede ayudarnos a encontrar el comando que deseamos.

```
nanday:~# pwd
/root
nanday:~# whoami
root
nanday:~#
```

Figura 2: Los comandos `pwd` y `whoami` nos dicen dónde estamos en el directorio y quiénes somos.

```

APT-GET (8)
NAME
  apt-get - APT package handling utility -- command-line interface
SYNOPSIS
  apt-get [-sqdyfmbw] [-o= config_string ] [-c= config_file ]
          [-t= { target_release_name | target_release_number_expression } ]
          {update | upgrade | dselect-upgrade | dist-upgrade |
          install pkg [ { =pkg_version_number | /target_release_name } ] ...
          | remove pkg... | purge pkg... |
          source pkg [ =pkg_version_number ] ... | build-dep pkg... |
          check | clean | autoclean | autoremove | {-v | --version} |
          {-h | --help}}
    
```

Figura 3: Las páginas man pueden ser intimidatorias en su meticulosidad, como muestra el inicio de la página para apt-get.

repetir el texto y pulsar el botón central (la rueda) del ratón.

## Navegación Básica

Cuando se es principiante en GNU/Linux, notamos enseguida que la estructura de directorios es completamente diferente a la de Windows. Probablemente aprenderemos rápidamente nuestro camino hacia el directorio de inicio, porque podemos hacer en él lo que nos plazca, aunque, más allá del hecho de que es un subdirectorio de /home, posiblemente tengamos sólo una ligera idea del resto del árbol de directorios.

Además, nuestro directorio de inicio está lleno de ficheros y directorios de configuración que comienzan por un punto y que normalmente no se listan cuando ejecutamos el comando `ls`. Éstos no se listan porque la mayoría del tiempo no es necesario verlos y, quizás, porque si están fuera de nuestra vista, es menos probable que interfiramos en ellos. El hecho de que no se listen no significa que no podemos usar el comando `cd` para cambiar a cualquiera de ellos. Si necesitamos ver sus nombres, entonces `ls -a` nos los revelará.

Si nos perdemos en la estructura de directorios mientras nos encontramos en la línea de comandos, podemos usar `pwd` (“present working directory”) para conseguir orientarnos. Si hemos estado cambiando cuentas y queremos asegurarnos, por ejemplo, de que no estamos ejecutando un comando peligroso como `root`, entonces el comando `whoami` nos dirá el nombre de la cuenta que estamos usando en este momento. Sin embargo, si olvidamos continuamente el nombre de la cuenta actual, podemos añadirla a nuestro prompt de la línea de comandos [1].

## Ahorro de Tiempo

Cuando introducimos los comandos, Bash tiene algunas formas de ahorrar tiempo y reducir la escritura. A menos

que se dé una ruta completa, cualquier fichero o subdirectorio que introduzcamos se supone que está en el directorio presente. Sin embargo, si deseamos estar absolutamente seguros (lo cual nunca es mala idea), podemos usar `./` para indi-

car el directorio presente. Por ejemplo, si estamos en el directorio de inicio de un usuario llamado *bruce*, entonces, introducir `/home/bruce/download` es lo mismo que escribir `./download`. Si nuestros subdirectorios se encuentran a distintos niveles de profundidad, este truco puede reducir de manera significativa lo que tenemos que escribir.

Existen algunos atajos diferentes que podemos usar con el comando `cd` (change directory):

- `cd` – (sin parámetros) nos mueve al directorio de inicio del usuario actual.
- `cd ..` – Nos mueve al directorio directamente encima del actual.
- `cd -` – Nos vuelve al directorio previo.
- `cd ~` – Nos vuelve al directorio de inicio para la cuenta actual.
- `cd ~/[nombre directorio]` – Cambia al directorio especificado dentro del directorio de inicio para la cuenta actual.

Estos dos últimos atajos explican por qué vemos a menudo referencias al directorio de inicio reducidas a una tilde. Por ejemplo, si vemos una referencia a `~/bashrc`, está indicando el fichero de configuración `.bashrc` en nuestro directorio de inicio.

## Búsqueda del Comando Ayuda

Cuando trabajamos en la línea de comandos, nos encontraremos indudablemente con comandos no familiares, muchos de ellos con muchas opciones que no podemos recordar fácilmente. Si todo lo que queremos es una explicación rápida de un comando, podemos usar `apropos` o `whatis`. El único problema es que, mientras estos comandos pueden servir como recordatorios para usuarios veteranos, su salida es a menudo enigmática para los menos experimentados. Por ejemplo, si introducimos `apropos chmod`, la definición que recibimos es “change file mode bits”, lo cual es lo suficientemente geeky como para espantar a los más principiantes (Figura 3).

Podemos conseguir información más detallada introduciendo `man` seguido por el nombre del comando. Sin embargo, la calidad de las llamadas páginas `man` varía mucho de unas a otras. A pesar de que algunas se leen con bastante facilidad, especialmente las más recientes, otras se encuentran escritas para una audiencia experta. De hecho, algunas han podido permanecer sin casi cambios durante una década o más.

En teoría, las páginas de información – las notas explicativas que recibimos cuando introducimos `info` seguido por el nombre de un comando – son más amigables que las páginas `man`. Y, en la práctica, así es. Sin embargo, a veces son sólo un poco mejor que las peores páginas `man` y pueden variar casi tanto en calidad. Además, el movimiento para que las páginas `info` sustituyan a las páginas `man` parece haberse debilitado en los últimos años.

A menudo, la ayuda más útil que podemos encontrar es usar la opción `-h` o `--help` para un comando, lo cual, normalmente, produce un resumen de opciones corto. Lo que falta, en mi opinión, es cualquier tipo de ayuda integrada que nos permita introducir una descripción de la acción que nos gustaría llevar a cabo y recibir una lista de comandos posibles. Desafortunadamente, las opciones disponibles son las que hay y la única otra es buscar en la web.

## Sólo el Principio

Naturalmente, queda mucho por detallar de Bash. Bash es un extenso capítulo al que dedicar años para conocer sus complejidades. De hecho, es tan amplio que poca gente puede presumir de conocerlo completamente.

Sin embargo, los puntos que se mencionan aquí deberían servir como orientación general y deberían ser suficientes para mostrar que Bash es increíblemente flexible, con incontables maneras de ahorrar esfuerzos – si nos tomamos el tiempo para aprender sobre él. Si nos familiarizamos con estas ideas básicas, podremos saber lo suficiente para aprovecharnos de sus utilidades. ■

**RECURSOS**

[1] “Tunea tu prompt” de Heike Jurzik, Linux Magazine – Edición en Castellano, número 35, página 85: [http://www.linux-magazine.es/issue/35/085-087\\_LdeCLM35.crop.pdf](http://www.linux-magazine.es/issue/35/085-087_LdeCLM35.crop.pdf)