

Examinando ficheros... de cerca

# HORIZONTES CERCANOS

Si desea echar un vistazo a un fichero de texto, Linux posee comandos en abundancia para satisfacer nuestra curiosidad.

**POR BRUCE BYFIELD**

**G**NU/Linux está diseñado para ser un sistema operativo práctico. Por esa razón, la mayoría de sus ficheros de configuración y registros del sistema se encuentran escritos en texto plano, lo que hace que sean fáciles de leer desde la línea de comandos. Si se desean modificar esos ficheros registrándonos como usuario root, deberemos usar un editor de textos como vi, emacs o nano. Aunque a menudo lo que desearemos no será realizar cambios, sino echarles un rápido vistazo para reunir información o ver si el sistema está operando tal y como debería.

Para ayudarnos a ver la información, GNU/Linux incluye algunos comandos de visualización. Para un vistazo breve en ficheros cortos, *cat* debería bastarnos. Sin embargo, para la mayoría de nuestros propósitos, probaremos con *more* o con *less*. Si estamos especialmente interesados en el comienzo

final del fichero, entonces usaremos *head* o *tail*. Lo básico de todos estos comandos es fácil de entender, aún más porque muchos usan opciones similares, o al menos funcionalidades similares.

## Dando de Comer al Gato

Su nombre cuenta la historia del propósito original del comando *cat*, forma corta de “concatenar”. En otras palabras, el comando está diseñado para unir ficheros. A pesar de que concatenar se encuentra listado todavía en numerosos resúmenes como el propósito principal, sospecho que la mayoría de la gente lo usa en la actualidad solamente para su segundo propósito de lectura de ficheros cortos.

Quizás la razón principal por la que *cat* sea tan popular se deba a que su sintaxis es muy fácil (Figura 1). Para ver un fichero, lo único que necesitamos hacer es introducir el comando

seguido del fichero que deseamos ver. Por ejemplo, si deseamos comprobar la lista de dispositivos montados en nuestro sistema, escribimos *cat /etc/fstab*.

Pero *cat* ofrece unas cuantas opciones más. Si hemos editado nuestra lista de repositorios en Debian o Ubuntu y hemos recibido un mensaje de error en una línea específica, podríamos contar las líneas manualmente, o – suponiendo que sabemos cómo – abrir */etc/apt/sources.list* en vi con las líneas enumeradas. Pero también podemos usar el comando *cat -b /etc/apt/sources.list* (Figura 2). Este comando enumera solamente las líneas con contenido e ignora las que estén en blanco. Si las líneas en blanco tienen importancia, podemos sustituir *-b* por *-n*. Otras opciones para *cat* marcan los caracteres de formato de tabulaciones, finales de línea y caracteres que no se imprimen. Además, tenemos

opciones para mostrar varias combinaciones de esos caracteres, aunque la mayor parte del tiempo, la única opción que probablemente necesitemos sea `-A`, que muestra todas ellas.

Aunque `cat` es fácil de usar, su principal desventaja es que no es apropiado para ficheros que tienen más líneas que las que nuestro terminal puede presentar en una pantalla única. Si tiene más, tenemos que desplazarnos arriba y abajo. Lo que es más importante, si un fichero tiene más líneas que el historial de nuestro terminal virtual, perderíamos la parte superior del fichero. A causa de estas desventajas, mucha gente prefiere usar una u otra de las opciones de vista.

### Más o Menos

El comando `more` es ligeramente más sofisticado que `cat` y más idóneo para ficheros de más de 20 líneas. Su principal virtud es que presenta información de pantalla en pantalla y elimina el riesgo de que perdamos parte de ella como puede perderse con `cat`.

La forma más simple de usar `more` es viendo un fichero directamente. Por ejemplo, un usuario llamado `allan` podría ver su correo del sistema introduciendo `more /var/mail/allan`.

A menudo veremos otro comando con una tubería que desvía la salida a través de `more`. Uno de los ejemplos más comunes de esto es un comando de búsqueda como `find` o `grep`, cuando sabemos que vamos a tener probablemente un número elevado de resultados. Por ejemplo, si deseamos localizar un número importante de gráficos PNG en el directorio actual y que los resultados sean filtrados a través de `more`, introduciríamos `find ".png" | more`.

Cuando los resultados llenan la pantalla, aparece el mensaje resaltado `--More` en la parte inferior de la lista. Para ir a la página de resultados siguiente pulsamos la barra espaciadora. Si consideramos que este mensaje básico no es suficiente, podemos usar la opción `-d` para expandirlo (Figura 3). Otras opciones útiles incluyen `+ <número>`, donde sustituimos `<número>` por el número de la línea a partir de la cual queremos comenzar a mostrar, y `+/<número>` `<cadena>` `<número>`, la cual

```
bruce@nanday:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
/dev/sda2 / reiserfs notail 0 1
/dev/sda9 /home reiserfs defaults 0 2
/dev/sda8 /tmp reiserfs defaults 0 2
/dev/sda5 /usr reiserfs defaults 0 2
/dev/sda6 /var reiserfs defaults 0 2
/dev/sda7 none swap sw 0 0
/dev/hda /media/cdrom0 udf,iso9660 user,noauto 0 0
/dev/scd0 /media/cdrom1 udf,iso9660 user,noauto 0 0
```

Figura 1: El comando `cat` es fácil de usar e idóneo para ver ficheros cortos.

```
bruce@nanday:~$ cat -b /etc/apt/sources.list
1 #Debian Stable
2 deb http://mirror.peer1.net/debian/ stable main
3 #deb ftp://ftp.debian.org/debian lenny main
4 #Debian Testing
5 deb http://mirror.peer1.net/debian/ squeeze main
6 deb ftp://ftp.debian.org/debian squeeze main
7 # deb-src http://mirror.peer1.net/debian/ squeeze main
```

Figura 2: Enumerar líneas con el comando `cat` puede ayudarnos a localizar errores.

muestra solamente líneas en las que ocurre `<cadena>`.

Una vez que `more` se está ejecutando, tenemos algunas opciones de navegación a través del teclado. Para buscar una cadena de texto, usamos `/"<cadena>`, o podemos presionar la barra espaciadora para presentar la pantalla de texto siguiente. Cuando estamos buscando, podemos pulsar `Enter` para avanzar una línea a la vez o introducir `z <número>` para especificar el número de líneas que queremos saltar hacia delante, además de establecer un nuevo número de líneas a avanzar para `z` sin argumentos (si pulsamos `z`, el predeterminado es `1`, a menos que lo hayamos cambiado). Cuando hemos acabado de ver, pulsamos `q` para volver al prompt de la línea de comandos.

Las opciones del comando `more` hacen que `cat` pueda parecer primitivo en comparación. Sin embargo, cuando usamos `more`, encontraremos que posee una desventaja considerable: No podemos desplazarnos hacia atrás, de manera que una pantalla desaparece una vez que se ha sobrepasado la parte superior del monitor, a menos que abandonemos `more` y reiniciemos. La verdad es que `more` se usa poco actualmente. La fecha última de su página man corresponde al 29 de

Junio de 1988, e incluso aquí hace referencia a `more` como “primitivo”. Hoy en día, el comando para ver el fichero que utiliza más gente es el más sofisticado `less`.

### Menos es Más

El nombre del comando `less` es una broma académica. Según ciertas teorías, “less es more” – es decir, la simplicidad es con frecuencia más elegante que la complejidad. El nombre del comando hace referencia a este dicho – y, en realidad, ¿qué mejor manera de indicar que fue escrito como un sustituto de `more`? De hecho, si uno está definido en nuestro entorno (tema que abordaremos otro día), existe incluso una variable `LESS_IS_MORE` que hace que cualquier intento de usar `more` sea sustituido por `less`.

El nombre se hace incluso más apropiado cuando aprendemos que `less` es una mejora de `more` en algunos sentidos. Aunque utiliza el mismo formato de comando que `more`, a diferencia de `more`, `less` nos permite volver hacia atrás. Además, ofrece más opciones, y como no lee el fichero de la entrada completo antes de presentar todo, es más rápido en iniciarse también.

Tanto `less` como `more` poseen algunas funciones comunes, aunque no

siempre usan las mismas opciones para acceder a ellos. Al igual que con *more*, podemos usar + <número> y +/"<cadena>" con *less* (como vimos antes), aunque los resultados de + /" <cadena>" se resaltan en la salida.

El comando *less* también posee una enorme cantidad de otras opciones. Algunas útiles para el usuario normal incluyen *-I*, que ignora la diferencia entre mayúsculas y minúsculas cuando estamos buscando una cadena específica y *-V*, que subraya tabulaciones, finales de línea y caracteres de no impresión. La opción *-w* resalta la primera línea después de que el texto se haya desplazado si hemos adelantado una sola línea, y *-W* resalta la primera línea cuando nos hemos adelantado más de una línea. Se advierte, sin embargo, que no todas estas opciones se encuentran disponibles en todas las versiones de Bash.

Los comandos de navegación son una mezcla parecida de nuevo y raro para veteranos de *more*. Ambos usan la barra espaciadora para avanzar la pantalla, aunque *less* nos permite adelantar la mitad de la pantalla también pulsando *d*. Además, podemos usar *b* con *less* para retroceder una pantalla completa, o *u* para retroceder media pantalla. De forma similar, podemos usar *z* <número> para especificar el número de líneas a adelantar, o *y* <número> para el número de líneas a retroceder. Las teclas de flechas RePg y AvPG navegan hacia adelante y hacia atrás, y las teclas de flechas hacia la izquierda y derecha mueven desde un extremo de una línea larga al otro extremo.

Si añadimos /" <cadena>" busca por resultados, igual que hace *more*, aunque añadiendo Ctrl+k al inicio de la cadena, resalta todas las instancias de la cadena (Figura 4). Si fuera necesario, podemos usar Esc+ u para eliminar el resaltado.

```
From root@nanday.nanday.com Wed Nov 28 00:07:26 2007
Return-path: <root@nanday.nanday.com>
Envelope-to: root@nanday.nanday.com
Delivery-date: Wed, 28 Nov 2007 00:07:26 -0800
```

Figura 4: El comando *less* puede resaltar resultados de la búsqueda.

```
nanday:/var/log# head -v ./syslog.0
==> ./syslog.0 <==
Jun 24 10:07:55 nanday syslogd 1.5.0#5: restart.
Jun 24 10:07:55 nanday anacron[3868]: Job `cron.daily' terminated
Jun 24 10:07:55 nanday anacron[3868]: Job `cron.weekly' started
```

Figura 5: Tanto el comando *tail* como *head* poseen modos *verbose* (salida más detallada) y *quiet*.

Otra funcionalidad práctica en *less* es su capacidad de trabajar con múltiples ficheros. Si usamos esta característica, entonces podemos navegar entre ellos con *:n* y *:p*. Al igual que con *more*, podemos usar *q* para cerrar el comando.

### ¿Cara o Cruz?

A pesar de que *less* es potente, a veces, al igual que *cat* y *more*, puede ser impreciso o excesivo para lo que deseamos. En muchas circunstancias, lo suyo es utilizar *head* o *tail* en su lugar.

Como su nombre implica, *head*, de manera predeterminada, examina las diez primeras líneas de un fichero. Probablemente, su mayor uso es permitarnos identificar un fichero rápidamente. Por ejemplo, si hubiéramos iniciado sesión como *root* y no estuviéramos seguros de la información que contiene */var/log/scrollkeeper.log*, *head* debería ser suficiente para encontrar esa respuesta.

*tail*, por el contrario, examina las diez últimas líneas de un fichero y las escribe a la línea de comandos. Es útil porque los sistemas de log generalmente escriben la última información en la parte inferior del fichero, lo que es ideal para la resolución de problemas. Todo cuanto necesitamos hacer es abrir un terminal en un escritorio virtual no usado y continuar nuestro trabajo en otro, ojeando ocasionalmente la salida de *tail* para ver qué está ocurriendo.

Si 10 líneas no son suficientes, usamos *-<número>* para cambiar el número de líneas que se presentan en *head* o *tail*. Si deseamos ver el encabe-

zamiento del fichero, el cual puede contener información como el nombre del fichero o información que identifica su formato, entonces utilizamos la opción *-v* (verboso) (Figura 5). Como alternativa, podemos suprimir el encabezado con *-q* (*quiet*).

Además, *tail* posee algunos comandos para ayudarnos a manejar la información relativa a la resolución de problemas. Si añadimos *-f*, la información se añadirá a la salida cuando esté disponible. El fichero monitorizado se vuelve inaccesible, podemos usar *--retry* para asegurarnos de que continuamos controlándolo (o para asegurarnos de que la razón por la que no hemos recibido nuevos datos no se deba a algo que haya pasado al fichero). Para controlar la frecuencia con la que el fichero es escaneado y reducir la cantidad de recursos del sistema utilizados por *tail*, añadimos *-s <segundos>*.

Otra funcionalidad útil es *<PID>*, la cual detiene la ejecución de *tail* si el proceso del sistema con la ID especificada detiene su ejecución. Lo más probable es que necesitemos ejecutar los comandos *top* o *ps* como usuario *root* para encontrar el PID del proceso que deseamos monitorizar.

### Un Comienzo Básico

Estas opciones no son las únicas que podemos usar con los comandos para la visualización de ficheros. En concreto, *less* posee una docena más de parámetros, algunos de los cuales probablemente no usaremos a menos que seamos administradores de sistemas avanzados. Aún así, lo mencionado aquí debería ayudarnos a usar los comandos con eficacia sin tener que enredarnos con demasiados detalles. Podemos echar un vistazo a la página man de cada comando para aprender más. Los comandos vistos no son complicados – incluso aunque a veces lo sean las opciones descritas en su página man. ■