Capítulo 5: Manejo de archivos

Introducción

La línea de comandos nos brinda acceso a varias herramientas que nos permiten examinar y procesar archivos de texto plano sin necesidad de abrirlos con un editor. Entre otras operaciones, esto incluye el poder *leerlos* (incluso cuando están comprimidos), *paginarlos* (para los más extensos), *mostrar las primeras* (o *últimas*) *líneas*, *contar la cantidad de líneas y caracteres* que incluyen, *comparar el contenido* de dos o más archivos, e incluso buscar patrones dentro de los mismos.

Además, al ir acumulando archivos, creando directorios, y luego instalando programas, resulta indispensable poder contar con una manera de *encontrarlos* en el sistema utilizando distintos criterios de búsqueda.

En este capítulo mostraremos cómo llevar a cabo todas estas operaciones. Podemos considerarlo como la primera muestra de la robustez de la línea de comandos y de las posibilidades que nos brinda.

---

*Una tarea habitual de todo **sysadmin** consiste en la inspección de **logs**, que son los registros de actividad del sistema. Otra de ellas es la revisión de archivos de configuración. El saber utilizar las herramientas que acabamos de mencionar es de suma utilidad para poder realizar ambas cosas eficazmente.*

El comando cat

Este comando concatena (*con**cat**enate*) archivos y los imprime en la salida estándar (por lo general se trata de la pantalla). Si no le pasamos ningún argumento, esperará a leer de la entrada estándar (el teclado). Por lo general solamente recurrimos a `cat` para visualizar archivos `cortos` únicamente.

Existe también `zcat` que hace lo mismo, pero con archivos comprimidos. Como la manera de utilizar estos dos comandos es idéntica, ilustraremos su uso empleando el primero.

Por ejemplo, el siguiente comando nos permitirá ver el contenido de los archivos indicados:

```
cat /etc/passwd /etc/group
```

Uso de more y less

Los comandos `more` y `less` paginan (dividen en páginas) uno o varios archivos y los muestran en la terminal. De no indicárseles un archivo, paginan la entrada estándar. Se diferencian en las facilidades que brindan como indicamos a continuación:

- `more` es más restrictivo en cuanto al movimiento dentro del texto, y visualiza sucesivamente el porcentaje del archivo examinado hasta el momento. Cuando se alcanza el final del último archivo a paginar, `more` termina automáticamente.
- Por otro lado, `less` cumple la misma función que `more`, pero además, nos permite realizar búsquedas de palabras dentro del contenido del archivo, resaltar los resultados, y saltar entre una ocurrencia y otra.

Por ejemplo:

```
less /etc/password
more /etc/passwd
```

Para examinar un archivo (o más) con cualquiera de estas dos herramientas:

- `q`: permite interrumpir el proceso y salir.
- `/criterio`: realiza búsquedas del patrón *criterio* dentro del texto.

Para realizar otra búsqueda empleando un patrón distinto sólo es necesario escribir / seguido de la palabra correspondiente.

- `[n]b` permite regresar *n* páginas (por defecto, n=1).
- `[n]f` es para adelantar *n* páginas.
- `h` para mostrar la ayuda disponible (salimos de la misma con la tecla `q`).

Además, en `less` podemos emplear

- `G` para desplazarnos al final del archivo.
- `g` para ir directamente al principio del archivo.

El comando `man` (que utilizamos para acceder a los *man pages* o manuales de los comandos), utiliza por defecto el paginador `less` para dar formato a su salida.

Existen además los comandos `zless` y `zmore` que permiten paginar a los archivos comprimidos sin necesidad de descompactarlos previamente en nuestro disco.

El comando `cat` puede ser interesante para ver el contenido de archivos pequeños, pero en general `less` resulta de mayor utilidad en la mayoría de los casos.

Principio y fin: head y tail

Los comandos `tail` y `head` muestran respectivamente el final y el comienzo (10 líneas por defecto) de uno o varios archivos. De no especificarse al menos un archivo toman la entrada estándar.

Ambos comandos tienen la misma sintaxis:

```
tail [opciones] [archivos]
head [opciones] [archivos]
```

*`tail` (que significa **cola**) es muy interesante para ver qué está sucediendo con un archivo en tiempo real. Con la opción `-f` nos muestra dicho archivo a medida que se le van agregando líneas.*

Otras opciones frecuentemente utilizadas de `tail` son:

- `tail -q`: coloca los encabezamientos con el nombre de los archivos cuando se indican varios (**quiet**).
- `tail -n`, seguido de un número, nos muestra esa cantidad de líneas desde el final del archivo en vez de las 10 líneas establecidas por defecto.

`tail` es uno de los comandos más usados por los usuarios por presentar la propiedad de mirar el archivo en el momento en que se están agregando datos al final.

Supongamos que queremos visualizar los eventos de autenticación del sistema, podemos hacer

```
tail -f /var/log/auth.log
```

Para poder apreciar la utilidad de este último comando será necesario que abramos otra terminal, e iniciemos otra sesión con la misma cuenta de usuario u otra. A continuación, deberíamos observar cómo se refleja ese evento en la primera terminal. Para finalizar la inspección y volver al *prompt*, presionemos Ctrl+c.

El comando wc

Una de las tareas más usuales de un *sysadmin* es la necesidad de contar la cantidad de líneas de un archivo o del resultado de un comando. Esto puede bien representar la cantidad de cuentas de usuario del sistema (una por línea en el archivo **/etc/passwd**), por nombrar un ejemplo.

Si no supiéramos cómo cumplir con el objectivo expuesto en el párrafo anterior, podríamos recurrir al `apropos` seguido de la palabra *counts* (de *contar* en inglés):

```
apropos counts
```

En la última línea del resultado (como podemos ver en la siguiente imagen) aparece nuestra mejor alternativa. El comando `wc` nos permite contar no solamente las líneas de un archivo sino también las palabras que contiene el mismo y hasta la cantidad de caracteres individuales presentes.

```
alumno@stretch:~$ apropos counts
get_avphys_pages (3) - get total and available physical page counts
get_phys_pages (3)   - get total and available physical page counts
wc (1)               - print newline, word, and byte counts for each file
alumno@stretch:~$
```

Una vez identificado el comando mediante el paso anterior, podemos consultar el *man page* de `wc` para ver cuáles son las opciones que debemos usar para contar las líneas de un archivo, las palabras, y la cantidad de bytes del mismo. En este caso se trata de `-l`, `-w`, y `-c`, respectivamente.

Dado el archivo **/etc/passwd**, los siguientes comandos nos permitirán ver los datos que se indican:

```
wc -l /etc/passwd # Contar líneas
wc -w /etc/passwd # Contar palabras
wc -c /etc/passwd # Contar bytes
```

Como hemos podido apreciar, el comando `apropos` nos permite hacer una búsqueda por palabra clave en la descripción de los comandos disponibles. Es sumamente útil para averiguar qué herramienta podemos utilizar para realizar una determinada tarea.

Otro uso clásico de este comando consiste en suministrarle, a través de una tubería, el resultado de un comando previo para que nos indique la cantidad de líneas presentes en dicho resultado. Volviendo a utilizar el mismo ejemplo que antes:

```
cat /etc/passwd | wc -l
```

también nos mostrará la cantidad de líneas, **pero solamente ese dato** (a diferencia de `wc -l /etc/passwd`, que además devuelve el nombre del archivo).

El comando diff

Dados los archivos de prueba **test1** y **test2**, el comando `diff` nos permite ver las diferencias entre ambos línea a línea.

Supongamos que **test1** es

```
Está en un laberinto de
pequeños pasajes sinuosos
que son todos iguales.
```

y **test2**:

```
Está en un laberinto de
pequeños pasajes sinuosos
que son todos diferentes.
```

El comando indicará las diferencias que ha encontrado, con el número de línea, y señalará (con un < y un >) en qué archivo se ha detectado la diferencia:

```
alumno@stretch:~$ diff test1 test2
3c3
< que son todos iguales.
---
> que son todos diferentes.
alumno@stretch:~$
```

Observemos cómo nos indica el comando `diff` que si elimina la línea "<" y agrega la línea ">", los dos archivos serían iguales.

Hermanos inseparables: locate y updatedb

Para realizar búsquedas rápidas de archivos en Linux disponemos de estas dos herramientas que se utilizan en conjunto. El propósito de `locate` es actualizar la base de datos en la que se mantiene la lista de todos los archivos del sistema. Para poder obtener resultados precisos, es necesario que dicha base de datos se mantenga actualizada, lo cual se lleva a cabo mediante la ejecución del comando `updatedb`.

La sintaxis básica de `locate` es la siguiente:

```
locate [OPCIONES] [PATRÓN]
```

donde si se utiliza la opción `-r` (o su equivalente `--regexp`), `PATRÓN` puede reemplazarse por una expresión regular. Otra opción muy útil es `-i` (sinónimo de `--ignore-case`), la cual ignora mayúsculas o minúsculas al examinar el `PATRÓN` dado.

Por defecto, cuando `locate` se utiliza sin opciones devolverá **TODOS** los archivos y directorios que **contengan el criterio de búsqueda en su ruta completa** (no solamente en el nombre del archivo propiamente dicho). De esta manera, `locate txt` se traduce en `locate *txt*` y regresa tanto **/usr/lib/firmware/qca/NOTICE.txt** como **/usr/bin/httxt2dbm**, por nombrar dos ejemplos.

Si se desea buscar por nombre exacto, podemos utilizar la opción `-b`, encerrar el criterio de búsqueda entre comillas simples y colocarle delante una barra invertida. Por ejemplo,

```
locate -b '\txt'
```

devolverá un objeto cuyo nombre sea exactamente igual a **txt**.

Veamos dos ejemplos adicionales en los que utilizamos expresiones regulares y anulamos la distinción entre mayúsculas y minúsculas al realizar la búsqueda:

```
locate -r usuarios[bB]ash.txt # Utilizar expresión regular
locate -i usuariosbash.txt # Ignorar mayúsculas y minúsculas
locate usuariosbash.txt # No devolverá nada
```

En la imagen que aparece a continuación se pueden apreciar los resultados:

```
alumno@stretch:~/clase$ ls
backupbasico.sh  ejemplocase.sh  for.sh  tareas.txt  usuariosBash.txt
alumno@stretch:~/clase$ locate -r usuarios[bB]ash.txt
/home/alumno/clase/usuariosBash.txt
alumno@stretch:~/clase$ locate -i usuariosbash.txt
/home/alumno/clase/usuariosBash.txt
alumno@stretch:~/clase$ locate usuariosbash.txt
alumno@stretch:~/clase$
```

En el último ejemplo de arriba, el comando no devuelve ningún resultado ya que por defecto, locate hará distinción entre mayúsculas y minúsculas.

Buscar con find

Si ya disponemos de `locate`, ¿cuáles son los motivos para hacer una búsqueda de archivos con `find`? Son muchas las razones por las que nos interesaría buscar archivos dentro de nuestro sistema, entre las cuales podríamos mencionar las siguientes.

Vamos a necesitar realizar una búsqueda de archivos con find cuando querramos:

1. trabajar con un archivo que no sabemos o no recordamos dónde se encuentra.
2. identificar archivos

- por propietario o grupo dueño.
- por fecha de modificación

- por permisos asignados
- por tipo de archivo (archivo regular, directorio, etc)
- combinando los criterios anteriores

La sintaxis de `find` para comenzar una búsqueda en el directorio especificado por `RUTA` es la siguiente:

```
find [RUTA] [OPCIONES]
```

Para buscar un archivo por nombre, utilizamos `find` seguido del directorio a partir del cual deseamos realizar la búsqueda, de la opción `-name` (o `-iname` si deseamos ignorar mayúsculas y minúsculas), y finalmente el nombre del archivo.

Si utilizamos un comodín (*) en el nombre del archivo como criterio de búsqueda, deberemos encerrar el mismo entre comillas para evitar que la shell lo expanda (lo cual sucedería **ANTES** de efectuar la búsqueda propiamente dicha).

Veamos los siguientes ejemplos:

- Buscar en el directorio **/etc** todos los archivos con extensión **.conf**:

```
find /etc -name '*.conf'
```

- Buscar los archivos cuyo tamaño esté entre 10 MB y 20 MB:

```
find / -size +10240k -size -20480k
```

- Buscar todos los archivos con find dentro del directorio **/etc** cuyo dueño sea el usuario root y que fueron modificados exactamente hace 2 meses (60 días):

```
find /etc -type f -user root -mtime 60
```

Si quisiéramos ver la misma lista de archivos que fueron modificados hace más de un cierto número de días, deberemos usar dicha cantidad con el signo + delante. En resumen:

- `-mtime x`: archivos modificados exactamente hace *x por 24* horas (lo cual se traduce en *x* cantidad de días).
- `-mtime -x`: modificados durante los últimos *x* días.
- `-mtime +x`: modificados hace más de *x* días.

- Especificar un permiso como criterio de búsqueda a través de la opción `-perm`:

```
find / -type f -perm 777 # Archivos con permisos 777 a partir de /

find . -type f -perm -o=x # Archivos ejecutables por todos en el
directorio actual
```

- Buscar archivos vacíos:

```
find / -type f -empty
```

Para ejecutar una acción determinada sobre los resultados de una búsqueda emplearemos la opción `-exec`. Para cambiar los permisos de 777 a 644, utilizaremos `-exec` seguida de la acción que deseamos realizar (en este caso `chmod 644` a todos los archivos resultantes, simbolizados por `{}+`:

```
find . -type f -perm 777 -exec chmod 644 {} +
```

Para borrar archivos usaremos la opción `-delete` de GNU `find` (en este caso no es necesario recurrir a `-exec` para especificar la acción deseada ya que la misma es provista por una opción en particular). Veamos cómo borrar archivos vacíos:

```
find . -type f -empty -delete
```

La diferencia entre `find` y `locate` es que el primero emplea la estructura del sistema de archivos (o una porción de la misma) para realizar una búsqueda, mientras que `locate` lee la ubicación de los archivos desde una base de datos interna.

Identificar recursos con whereis

Al realizar las búsquedas mencionadas con `whereis` podemos especificar qué estamos necesitando. Si nos interesa encontrar los binarios utilizaremos la opción `-b`. Por otro lado, si queremos identificar el archivo correspondiente al *man page*, usaremos `-m`. Finalmente, con `-s` indicaremos que deseamos ubicar el código fuente. Si se omiten estas opciones, `whereis` devolverá los tres recursos mencionados simultáneamente. También podemos combinar dos de ellas para devolver la información correspondiente.

Los siguientes ejemplos nos servirán para ilustrar el uso de este comando:

```
# Todas las opciones
whereis top
# Solamente el binario
whereis -b top
# El man page
whereis -m top
# Buscar fuentes
whereis -s top
```

Veamos el resultado:

```
alumno@stretch:~$ whereis top
top: /usr/bin/top /usr/share/man/man1/top.1.gz
alumno@stretch:~$ whereis -b top
top: /usr/bin/top
alumno@stretch:~$ whereis -m top
top: /usr/share/man/man1/top.1.gz
alumno@stretch:~$ whereis -s top
top:
alumno@stretch:~$
```

En resumen, `whereis` es muy útil para buscar comandos ya que nos brinda información adicional. También nos muestra desde donde se configura si es que es configurable y donde está la documentación.

El comando which

Al pasarle como primer argumento un comando `which` nos dirá, de los directorios que existen en `$PATH`, en cuál de ellos está. Por ejemplo, `which ls` nos debería devolver `/bin/ls`.

Dos detalles interesantes sobre el uso de `which` son los siguientes:

- Una vez que se identifica un binario dado en un directorio incluido en `$PATH`, la búsqueda se detiene a menos que se utilice la opción `-a`. En ese caso se buscará en **TODOS** los directorios en vez de finalizar con la primera ocurrencia.
- Si uno o más de los argumentos no se encuentra en `$PATH`, `which` regresa un *exit status* de 1. Si todos se encuentran, el *exit status* será igual a 0, mientras que será igual a 2 si se especifica una opción no válida.

Filtros y búsqueda de patrones con grep

El comando `grep` (**G**lobally **R**egular **E**xpressions **P**attern) busca patrones en archivos. Por defecto devuelve todas las líneas que contienen un patrón determinado en uno o varios archivos. Utilizando las opciones se puede variar mucho este comportamiento. Si no le pasamos ningún archivo como argumento hace la búsqueda en su entrada estándar.

Este comando tiene la siguiente sintaxis:

```
grep [OPCIONES] [PATRÓN] [ARCHIVOS]
```

Las opciones más utilizadas de este comando son:

- `grep -c`: devuelve sólo la cantidad de líneas que contienen al patrón.
- `grep -i`: ignora las diferencias entre mayúsculas y minúsculas.

- `grep -H`: imprime además de las líneas, el nombre del archivo donde se encontró el patrón. Es así por defecto cuando se hace la búsqueda en más de un archivo.
- `grep -l`: cuando son múltiples archivos sólo muestra los nombres de aquellos donde se encontró al patrón y no las líneas correspondientes.
- `grep -v`: devuelve las líneas que no contienen el patrón.
- `grep -r`: busca en un directorio de forma recursiva.
- `grep -n`: imprime el número de cada línea que contiene al patrón.

Los siguientes ejemplos nos servirán para ilustrar el uso de este comando tan robusto:

- Identifica dentro de los archivos contenidos en **/usr/share/doc** en forma recursiva las líneas que contienen la palabra **linux**:

```
grep -H -r linux /usr/share/doc
```

- Busca dentro del archivo **/var/log/messages** las líneas que contienen la palabra **log** y nos dice cual es dicha línea:

```
grep -n log /var/log/messages
```

- Busca al usuario llamado **fabian** dentro de **/etc/passwd**:

```
grep -i fabian /etc/passwd
```

- Busca la cantidad de veces que aparece la palabra **root** en el archivo **/etc/group**:

```
grep -c root /etc/group
```

- Busca la coincidencia **mess** dentro de los archivos del directorio **/var/log** y nos muestra el nombre del archivo que la contiene:

```
grep -l -r -i mess /var/log
```