

Capítulo 5

MANEJO DE ARCHIVOS

Manejo de archivos de texto

- 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.
- Todas las habilidades mencionadas son esenciales para un sysadmin.

El comando cat

- Este comando concatena (*con**cat**enate*) archivos y los imprime en la salida estándar
- Si no le pasamos ningún argumento, esperará a leer de la entrada estándar
- Por lo general solamente recurrimos a **cat** para visualizar archivos cortos únicamente
- Existe también **zcat** que hace lo mismo, pero con archivos comprimidos (nos permite leerlos sin necesidad de descomprimirlos primero)

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

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

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.
- 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.

more y less (Cont.)

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

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

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

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 -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. Si hacemos **head -n** seguido de un número podremos ver las primeras líneas del archivo.
- **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.

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

El comando wc

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
```

- 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:

```
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.
- 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:

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.
```

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

Búsquedas rápidas

- Utilizamos **updatedb** y **locate** 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.

Búsquedas precisas con find

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

Búsquedas precisas con find (Cont.)

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.

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.

Búsquedas precisas con find (Cont.)

- 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
```

- 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
```

Búsquedas precisas con find (Cont.)

- 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
```

El comando whereis

- Este comando nos permite ubicar en el sistema los archivos correspondientes al archivo binario, al *man page*, y al código fuente relacionados con un comando dado.
  - 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.

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

```
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:~$
```

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.

El comando 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.

El comando grep (Cont.)

- **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.

Ahora veamos algunos ejemplos…

El comando grep (Cont.)

- 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
```