ShellScript

Gestión de ficheros y directorios

Al igual que en otros sistemas operativos, como Microsoft Windows, tenemos disponibles una serie de comandos básicos y de gestión de sistemas de ficheros. Algunos de ellos son más avanzados, puesto que tocan algunos aspectos, como los permisos, que no son sencillos a primera vista.
Así que vamos a centrarnos en los comandos más simples de gestión de ficheros y directorios, así como en comandos básicos que utilizaremos en nuestro día a día de trabajo con la shell.

Mostrar directorios y ficheros (ls)

El **comando cd** (Change Directory) nos permitirá cambiar el directorio actual o directorio de trabajo.
Si no se le pasa parámetro, el directorio cd cambia el directorio actual al directorio personal del usuario actual.
Si se le pasa parámetro, espera la ruta del nuevo directorio actual, ya sea relativa o absoluta.
El carácter ~ es un carácter especial, que tiene importancia en bash. Se trata de una variable que bash sustituye por la ruta del directorio personal del usuario actual.
Podemos emplear un punto para representar el directorio actual, y dos puntos para representar el directorio padre.

El **comando pwd** (Print Working Directory) nos muestra la ruta del directorio actual.

El **comando ls** (List) sin parámetros, mostrará el contenido del directorio actual.
Si le pasamos parámetros (que podemos combinar o separar) tendremos distintas opciones:

- **ls -l** nos mostrará información extendida de los ficheros.
- **ls -l -h** nos mostrará información extendida de los ficheros y el tamaño lo formateará.
- **ls -R** nos mostrará información de los directorios de forma recursiva.
- **ls -t** muestra ordenadamente los ficheros de más nuevos a viejos.
- **ls -d** nos muestra sólo los directorios

Más información: enlace

Obtener la salida de un comando

La sintaxis es: $(comando)
echo "los permisos del fichero datos.txt son $(ls -l datos.txt \| cut -d" " -f1)

Y si queremos guardar el resultado de un comando:
variable=$(comando)
echo "La salida del comando es: $variable"

Crear directorios (mkdir)

El **comando mkdir** (make directory) crea uno o más directorios. Aquellos cuya ruta se pase por parámetro.
Crear uno: ***mkdir directorio***
Crear varios: ***mkdir {directorio1, directorio2}***
Combinando ambas técnicas para crear subdirectorios: ***mkdir -p directorio1/{subdirectorio1, subdirectorio2}***

Más información: enlace

Crear ficheros (touch)

Hay varias formas de crear ficheros, la que vamos a ver es mediante el **comando touch**, solamente ponemos el comando y el nombre del ficheros.
touch archivo.txt

Eliminar directorios y ficheros (rm)

El **comando rmdir** (remove directory) borrará directorios, sólo si están vacíos. En caso de no estar vacío, deberemos borrar recursivamente su interior mediante el parámetro -r y el **comando rm**, que elimina todos los ficheros y directorios. Con el parámetro -i se preguntará antes de eliminar.
rm -r directorio

Copiar directorios y ficheros (cp)

Con el comando cp (copy) podremos copiar tanto ficheros como directorios.
cp directorio1 directorio2
El directorio 2 será una copia del directorio 1 y se crea en la ruta actual a menos que se indique otra.

Mover o renombrar directorios y ficheros (mv)

El comando mv se utiliza para mover o renombrar los archivos y directorios en su sistema de archivos del sistema operativo.
Cambiar el nombre, deberemos indicar el fichero y luego, en su misma ruta, el nuevo nombre:
mv fichero.txt ficheroNuevoNombre.txt
Mover de un origen a un destino:
mv rutaOrigen/archivo rutaDestino
Mover todos los archivos de un tipo a otra carpeta
mv *.sh carpetaDestino

Más información: enlace

Mostrar contenido de un fichero (cat)

Para ello, se suele utilizar el **comando cat** (conCATenate files), que técnicamente se considera un comando que concatena (une) la salida de uno o varios ficheros. Esto, dicho de otra forma, es que cat no se limita a mostrar el contenido de un solo fichero (como suele usarse habitualmente), sino que puede hacer cosas aún más potentes.
Por ejemplo, mostramos el fichero /etc/paswd, que almacena información de la cuenta del usuario (ya lo veremos en otro apartado en profundidad).
cat /etc/passwd
Un parámetro interesante de cat es **-s** que suprime espacios consecutivos repetidos, convirtiéndolos en uno sólo, o **-n** que numera la líneas a la izquierda.

También podemos utilizar cat para crear y agregar datos a un fichero, para ello hacemos uso de:
> para redirigir el contenido de un fichero a otro, o en caso de no existir, crear ese fichero.
>> para agregar datos a un fichero

Si queremos ver la última parte de un fichero o el principio utilizaremos los siguientes comandos:
El **comando tail**, por defecto muestra las últimas 10 líneas.
tail fichero
Para elegir el número de líneas usamos el parámetro **-n** seguido del número.
tail -n1 fichero

El **comando head**, por defecto muestra las primeras 10 líneas.
head fichero

Buscar ficheros (find)

El **comando find** se encarga de realizar búsquedas de ficheros y carpetas.
Su sintaxis es: find [ruta] [parámetros] [valores]

Ejemplos:
Buscar un fichero concreto:
find /home/david -name examen.txt
Si queremos ignorar mayúsculas y minúsculas:
find /home/david -iname examen.txt
Para buscar directorios:
find /home/david -type d -name ejercicios

Para buscar archivos de un tipo en concreto:
find /home/david -type f -name *.sh
Para buscar archivos de un tipo u otro:
find /home/david -name "*.js" -or -name "*.css" -type f
Buscar y borrar:
find /home/david -type f *.sh -delete

Más información: enlace

Buscar en ficheros (grep)

El **comando grep** busca un patrón que definamos en un archivo de texto, es decir, mediante su uso puedes buscar una palabra o patrón y se imprimirá la línea o líneas que la contengan.
A primera vista, puede parecer un comando poco útil, sin embargo, los administradores de sistemas que manejan muchos servicios con varios archivos de configuración lo usan para consultar o buscar líneas específicas dentro de esos archivos.
Su sintaxis es: ***grep [opciones] pattern [archivo]***

Sus opciones más importantes son:
grep -i para que no distinga entre mayúsculas y minúsculas
grep -c solo mostrará el número de líneas que coinciden con el patrón buscado
grep -r habilita la búsqueda recursiva en el directorio actual
grep -n busca líneas y precede cada línea coincidente con un número de línea
grep -v muestra las líneas que no coinciden con el patrón que hemos buscado

En el siguiente ejemplo, se imprimirán las líneas del archivo “contraseñas.txt” que contengan la palabra clave sin tener en cuenta mayúsculas y minúsculas:
grep -i clave contraseñas.txt

Más información: enlace

Cortar en un fichero (cut)

El **comando cut** se encarga de cortar las columnas o campos seleccionados de uno o más ficheros. Entre sus usos más habituales se encuentra sacar información de los campos o simplificar ficheros más complejos. Se utiliza de manera asidua en la elaboración de scripts, sobre todo para filtrar un resultado de una consulta o la combinación de un conjunto de comandos.
Su sintaxis es: ***cut -d [delimitador] -f [campos]***

Ejemplos:
Si queremos obtener de una dirección IP, el identificador del host (la última parte).
192.168.1.27
Vemos que están separados por un punto, el cual será nuestro delimitador. Tras hacerlo, tendremos cuatro columnas o campos, y la que nos interesa es la última, la del host (podemos indicar más):
cut -d "." -f 4
Este comando nos dará 27
cut -d "." -f 1,2 nos dará 192 y 168

Más información: enlace

Ordenar un fichero (sort)

El **comando sort** se usa para ordenar las líneas de un archivo, en un orden particular. Los espacios en blanco son tomados por defecto como separadores de campo.
Su sintaxis es: sort [opciones] [archivo]

Y algunos parámetros que podemos usar son:
sort -r invertirá el orden
sort -n toma un valor alfabético y lo interpreta como un número
sort -f ignora mayúsculas y minúsculas
sort -t se utiliza para especificar el separador de campo
sort -k busca según el número de columna y lo ordena

Ejemplos:

Ordenar el contenido de un fichero alfabéticamente (toma la primera letra de cada línea)

sort fichero.txt

En caso de que las líneas empiecen con números, podemos utilizar lo siguiente:

sort -n fichero.txt

Más información: enlace

Traducir y eliminar carácteres (tr)

El **comando tr** (traslate) permite traducir y eliminar carácteres, convertir mayúsculas a minúsculas, juntar caracteres repetidos…
Sintaxis: tr [opciones] “Conjunto1” “Conjunto2”

Ejemplos:
Nos puede interesar cambiar todas las comas por espacios en blanco:
tr “,” “ “
O eliminar espacios en blanco con más de uno.
tr -s “ “
O buscar y eliminar todos los carácteres que indiquemos:
tr -d “.”

Más información: enlace

Eliminar duplicados (uniq)

El **comando uniq**, es una herramienta de línea de comando que se utiliza para informar u omitir cadenas o líneas repetidas. Esto básicamente filtra líneas coincidentes adyacentes de INPUT (o entrada estándar) y escribe en OUTPUT (o salida estándar). Sin opciones, las líneas coincidentes se fusionan con la primera aparición.
Sintaxis: uniq [opciones] archivo

Algunos parámetros:
-c contar las líneas duplicadas
-i ignorar mayúsculas y minúsculas
-u para ver líneas que no se repiten

Ejemplos:
Omitir duplicados de un fichero
uniq fichero

Mostrar el número de líneas repetidas
uniq -c fichero

Más información: enlace

Contar líneas (wc)

El **comando wc** (word count) permite realizar diferentes conteos desde la entrada estándar, ya sea de palabras, caracteres o saltos de líneas.
Sintaxis: wc [parámetros] fichero

Parámetros
wc -l <fichero> número de líneas
wc -m <fichero> imprime el número de caracteres
wc -w <fichero> imprime el número de palabras

Ejemplo:
wc examen.txt

Más información: enlace

Comparadores en el if

Tenemos parámetros para utilizar con ficheros:

-d te permitirá saber si es un directorio y si existe
-f lo mismo que en el caso anterior pero para archivos
-e para ver si existe el archivo
-r en este caso te permite saber si el archivo tiene permiso de lectura
-s con esta opción puedes saber si el tamaño del archivo es mayor que cero. Es decir, que no se trata de un archivo vacío
-w te permitirá identificar si el archivo tienen permisos de escritura
-x lo mismo que en el caso anterior pero para el caso de permisos de ejecución.

Ejemplos:

Comprobar que un directorio existe

```
if [[ -d $directorio ]]; then
        echo “$directorio existe”
else
        echo “$directorio no existe”
fi
```

Comprobar que un fichero es de lectura **y** escritura

```
if [[ -r $fichero ]] && [[ -w $fichero  ]]; then
        echo “El fichero $fichero tiene permisos”
else
        echo “El fichero $fichero no tiene permisos”
fi
```

Leer línea a línea un fichero

```
while read linea; do
        echo $linea
done < $fichero
```

Permisos de ficheros y directorios

Para linux es muy importante la seguridad de la información de sus usuarios. Para preservar esa información, linux utiliza un sistema de permisos y privilegios que pueden definirse o asignarse a algunos usuarios y quitarse o restringirse a otros. De este modo se evita que un usuario elimine o modifique la información de otro.

Por eso en la estructura de directorios, hay un directorio para cada usuario y cada usuario solo puede modificar la estructura de directorios y archivos a partir de allí (de su directorio home).
Además, linux también gestiona grupos de usuarios y permite definir o asignar y quitar o restringir permisos para cada grupo de usuarios.

Esto da la posibilidad de configurar la seguridad de cada archivo, directorio o incluso servicios, aplicaciones y dispositivos en dos niveles, es decir, para cada usuario y para cada grupo de usuarios.
Básicamente, en linux podemos determinar quien puede hacer determinadas cosas y quien no.
El usuario “root”, es quien tiene todos los permisos sobre el sistema y es capaz de hacer lo que desee.

En linux, quien crea un archivo o un directorio es el propietario de esa información y por lo tanto, puede restringirle el acceso a ella a cualquier otro usuario salvo al root. Por defecto, los miembros de tu mismo grupo, no pueden modificar tu información, pero tú puedes darles permiso para hacerlo (o el usuario root).

En este tema nos centraremos en ver los permisos actuales de los ficheros, que podemos hacerlo con:
ls -l

Ejemplo:
-rwxr-xr-x …

El primer carácter indica si es un archivo (-), directorio (d) o link (l).
Del segundo al cuarto carácter, indican los permisos del propietario.
Del quinto al séptimo carácter, indican los permisos del grupo del propietario.
Del octavo al décimo carácter, indican los permisos del resto de usuario.

De Lectura (se representa con la letra r por **r**ead).
De Escritura (se representa con la letra w por **w**rite).
De Ejecución (se representa con la letra x por e**x**ecute).

Fuente y más información: enlace