Práctica de laboratorio: Navegar por el sistema de archivos y la configuración de permisos de Linux

Instrucciones

Parte 1: Explorar sistemas de archivos en Linux

El sistema de archivos de Linux es una de sus características más populares. Si bien Linux admite muchos tipos diferentes de sistemas de archivos, aquí nos enfocaremos en la familia **ext**, uno de los sistemas de archivos más comunes en Linux.

Paso 1: Accedan a la línea de comandos.

Abran la una VM y, luego, ingrese el usuario y contraseña en la consola

Configurar el teclado de acuerdo a su uso y costumbre.

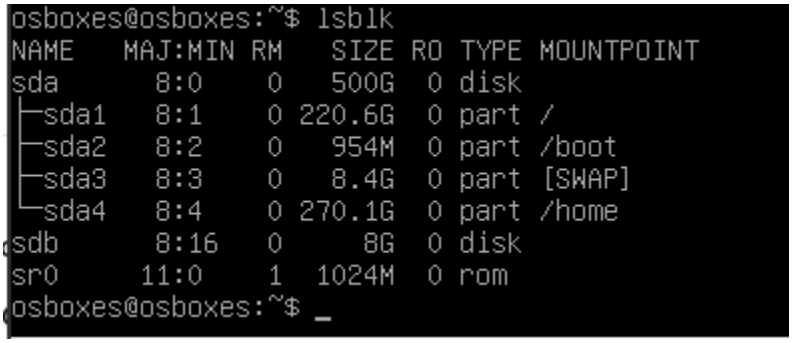
```
Unset
$ sudo dpkg-reconfigure keyboard-configuration
```

En mi caso es español latinoamericano.

Paso 2: Muestren los sistemas de archivos montados en este momento.

Los sistemas de archivos se deben *montar* primero para poder acceder a ellos y utilizarlo. En informática, *montar un sistema de archivos* significa tomar las medidas necesarias para que el sistema operativo pueda acceder a él. El montaje de un sistema de archivos es el proceso de vincular la partición física en el dispositivo de bloques (disco duro, unidad SSD, pen drive, etc.) a un directorio, a través del cual se puede acceder a todo el sistema de archivos. Como el directorio antes mencionado pasa a ser la raíz del sistema de archivos recién montado, también se lo conoce como *punto de montaje*.

a. Utilicen el comando **lsblk** para mostrar todos los dispositivos de bloques:

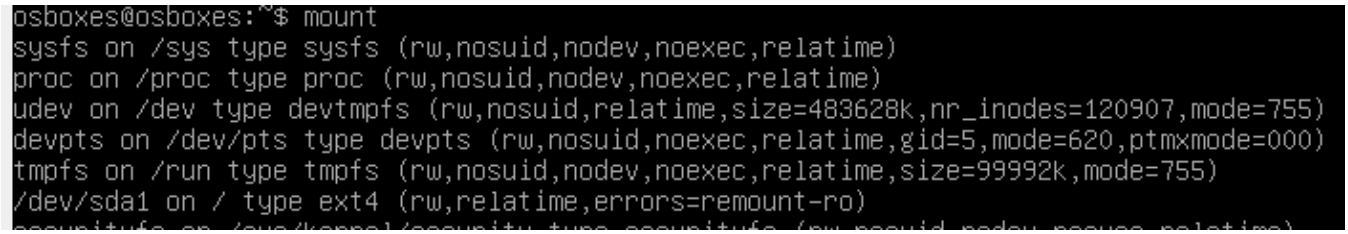


En la salida anterior se muestra que la VM tiene tres dispositivos de bloques instalados: sr0, sda y sdb. La salida en forma de árbol también muestra las particiones de sda y sdb. Convencionalmente, Linux utiliza /dev/sdX para representar discos duros, y el número del final representa el número de partición dentro de ese dispositivo. En las computadoras con varios discos duros probablemente se verán más dispositivos /dev/sdX. Si Linux se estuviera ejecutando en una computadora con cuatro discos por

ejemplo, los mostraría como /dev/sda, /dev/sdb, /dev/sdc y /dev/sdd de manera predeterminada. La salida implica que sda y sdb son discos duros, y que cada uno contiene una sola partición. En la salida también se muestra que sda es un disco de 500GB mientras que sdb tiene 8GB.

Nota: A menudo, Linux también muestra las unidades Flash USB como /dev/sdX, dependiendo de su tipo de firmware.

b. Utilicen el comando **mount** para mostrar información más detallada sobre los sistemas de archivos montados en este momento en la VM .



Muchos de los sistemas de archivos anteriores están fuera del alcance de este curso y son irrelevantes para la práctica de laboratorio. Nos enfocaremos en el *sistema de archivos raíz*, el que está almacenado en **/dev/sda1**. El sistema de archivos raíz es el sitio en el que se almacena el sistema operativo Linux en sí; todos los programas, las herramientas y los archivos de configuración se almacenan en el sistema de archivos raíz de manera predeterminada.

c. Volver a ejecutar el comando **mount**, pero esta vez, agregar el pipe | para enviar el resultado de mount a **grep** para filtrar el resultado y solo mostrar el archivo de sistema root:

```
[osboxes@publicserver ~]$ mount | grep sda1
```

En la salida filtrada anterior, el montaje nos indica que el sistema de archivos raíz está ubicado en la primera partición del dispositivo de bloques sda (/dev/sda1). Sabemos que es el sistema de archivos raíz gracias al punto de montaje utilizado: "/" (el símbolo de la barra diagonal). La salida también nos informa el tipo de formato que se utilizó en la partición: ext4, en este caso. La información entre paréntesis está relacionada con las opciones de montaje de particiones.

d. Emitan los siguientes dos comandos en la **VM** :

```
[osboxes@publicserver ~]$ cd /
[osboxes@publicserver /]$ ls -l
```

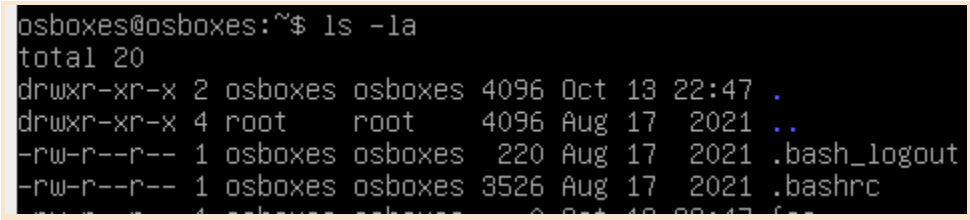
¿Cuál es el significado de la salida? ¿Dónde se almacenan físicamente los archivos de la lista?

¿Por qué no se muestra **/dev/sdb1** en la salida de arriba?

Paso 3: Montaje y desmontaje manual de sistemas de archivos

El comando **mount** también se puede utilizar para montar y desmontar sistemas de archivos. Tal como se ve en el Paso 1. La VM tiene dos discos duros instalados. El kernel reconoció al primero como /dev/sda y al segundo como /dev/sdb. Para poder montar un dispositivo de bloques, este debe tener un punto de montaje.

a. Utilicen el comando **ls -l** para verificar que el directorio **second_drive** sea el directorio de inicio del analista.



utilice el comando **mkdir second_drive** para crearlo.

```
[osboxes@publicserver ~]$ mkdir second_drive
```

b. Vuelvan a utilizar **ls -l** para generar una lista con el contenido del directorio second_drive recién creado.

```
[osboxes@publicserver ~]$ ls -l second_drive/
total 0
```

Observen que el directorio está vacío.

c. Utilicen el comando **mount** para montar **/dev/sdb1** en el directorio **second_drive** recién creado. La sintaxis del montaje es la siguiente: **mount [opciones] <dispositivo que se debe montar> <punto de montaje>**.

```
[osboxes@publicserver ~]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] contraseña para osboxes:
```

No se genera ninguna salida; eso quiere decir que el proceso de montaje tuvo éxito.

d. Ahora que se ha montado **/dev/sdb1** en **/home/analyst/second_drive**, utilicen **ls -l** para volver a generar una lista con el contenido del directorio.

```
[osboxes@publicserver ~]$ ls -l second_drive/
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-r--r-- 1 root root 183 Mar 3 15:42 myFile.txt
```

¿Por qué ya no está vacío el directorio? ¿Dónde se almacenan físicamente los archivos de la lista?

e. Emitan el comando **mount** sin opciones nuevamente para mostrar información detallada sobre la partición **/dev/sdb1**. Como antes, utilicen el comando **grep** para mostrar solamente los sistemas de archivos de **/dev/sdX**:

```
[osboxes@publicserver ~]$ mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime)
/dev/sdb1 on /home/osboxes/second_drive type ext4 (rw,relatime)
```

f. Desmontar sistemas de archivos es igual de simple. Recordar cambiar de directorio a una ubicación fuera del punto de montaje y utilicen el comando **umount** tal como se indica a continuación:

```
[osboxes@publicserver ~]$ sudo umount /dev/sdb1
[osboxes@publicserver ~]$
[osboxes@publicserver ~]$ ls -l second_drive/
total 0
```

g. Investigar: Es posible asignar permisos específicos al montar un sistema de archivos ? ¿Puede dar un ejemplo ?

Parte 2: Permisos de archivo

Los sistemas de archivos de Linux tienen características integradas para controlar la capacidad que tienen los usuarios de ver, cambiar, navegar y ejecutar el contenido del sistema de archivos. Esencialmente, cada archivo de los sistemas de archivos lleva consigo su propio conjunto de permisos, siempre con un conjunto de definiciones sobre lo que los usuarios y grupos pueden hacer con el archivo.

Paso 1: Visualizar y cambiar los permisos de archivo

a. Diríjanse a /home/usuario

```
[osboxes@publicserver ~]$ cd /home/usuario
```

b. Utilicen el comando **ls -l** para mostrar los permisos de archivo.

```
[osboxes@publicserver]$ ls -l
total 0
```

c. El comando **touch** es muy simple y útil. Permite crear rápidamente un archivo de texto vacío. Utilicen el comando **touch** para crear un archivo y vuelva a mostrar los permisos los permisos de archivo.

```
[osboxes@publicserver]$ touch foo
[osboxes@publicserver]$ ls -l
```

Consideren el archivo **foo** a modo de ejemplo. ¿Quién es el propietario del archivo? ¿Y del grupo?

Los permisos para **foo** son **–rw-r--r--**. ¿Qué significa esto?

d. Utilicen el siguiente comando para crear un archivo vacío en el directorio **/mnt**:

```
[osboxes@publicserver scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
```

¿Por qué no se creó el archivo? Genere una lista con los permisos, la propiedad y el contenido del directorio **/mnt** y explique qué sucedió. Cuando se agrega la opción **-d**, muestra el permiso del directorio principal. Registren las respuestas en las siguientes líneas.

```
[osboxes@publicserver ~]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan 5 2018 /mnt
```

¿Qué se puede hacer para que el comando **touch** que se muestra arriba tenga éxito?

e. El comando **chmod** se utiliza para cambiar los permisos de un archivo o directorio. Como antes, monten la partición **/dev/sdb1** en el directorio **second_drive** que ya se creó en esta práctica de laboratorio:

```
[osboxes@publicserver ~]$ sudo mount /dev/sdb1 ~/second_drive/
```

f. Pasen al directorio **second_drive** y generen una lista con su contenido:

```
[osboxes@publicserver ~]$ cd ~/second_drive
[osboxes@publicserver second_drive]$ ls -l
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-r--r-- 1 root root 183 Mar 3 15:42 myFile.txt
```

¿Cuáles son los permisos del archivo **myFile.txt**?aquí.

g. Utilicen el comando **chmod** para cambiar los permisos de **myFile.txt**.

```
[osboxes@publicserver second_drive]$ sudo chmod 665 myFile.txt
[osboxes@publicserver second_drive]$ ls -l
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-rw-r-x 1 root root 183 Mar 3 15:42 myFile.txt
```

¿Cambiaron los permisos? ¿Cuáles son los permisos de **myFile.txt**?

El comando **chmod** toma permisos en formato octal. De esa manera, el desglose del 665 es el siguiente:

6 en octal es 110 en binario. Suponiendo que cada posición de los permisos de un archivo puede ser 1 o 0, 110 significa rw- (leer=1, escribir=1 y ejecutar=0).

Por lo tanto, el comando **chmod 665 myFile.txt** cambia los permisos a:

Owner: rw- (6 in octal or 110 in binary)

Group: rw- (6 in octal or 110 in binary)

Other: r-x (5 in octal or 101 in binary)

¿Qué comando cambiaría los permisos de myFile.txt a rwxrwxrwx, con lo que se otorgaría acceso total al archivo a cualquier usuario del sistema?

h. El comando **chown** se utiliza para cambiar la titularidad de un archivo o directorio. Emitir el comando de abajo para hacer a root propietario de **myFile.txt**:

```
[osboxes@publicserver second_drive]$ sudo chown osboxes myFile.txt
[osboxes@publicserver second_drive]$ ls -l
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-rw-r-x 1 osboxes root 183 Mar 3 15:42 myFile.txt
[osboxes@publicserver second_drive]$
```

Nota: Para cambiar ambos, propietario y el grupo a **osboxes** al mismo tiempo, utilice el **sudo chown osboxes:osboxes myFile.txt** .

i. Ahora que el usuario analyst es el propietario del archivo, anexen la palabra ‘test’ (prueba) al final de **myFile.txt**.

```
[osboxes@publicserver second_drive]$ echo test >> myFile.txt
[osboxes@publicserver second_drive]$ cat myFile.txt
```

¿Fue exitosa la operación? Explique.

Paso 2: Directorios y permisos

En forma similar a lo que sucede con archivos comunes, los directorios también tienen permisos, tanto los archivos como los directorios tienen 9 bits para los permisos del propietario/user, group y otros. También hay tres bits más para permisos especiales: setuid, setgid y sticky que está más allá del alcance de este laboratorio.

a. Regresen al directorio home y emitan el comando **ls -lR** para generar una lista de todos los archivos con detalles:

```
[osboxes@publicserver second_drive]$ cd ~
```

Comparen los permisos del directorio con los del archivo . ¿Cuál es la diferencia entre la parte inicial de ambos?

Los comandos **chmod** y **chown** funcionan con los directorios del mismo modo que con los archivos.

Ejemplo Apache /var/www

Uno de los primeros pasos al instalar un servidor apache es asegurar los permisos de la carpeta /var/www.

En este sitio encontraremos información sobre el proceso con lujos de detalle.

https://usuariodebian.blogspot.com/2020/09/apache-permisos-usuario-y-grupo-www-data.html

Reflexión

Los permisos y la titularidad de los archivos son dos de los aspectos más importantes de Linux. También son una causa común de problemas. Un archivo con los permisos o la titularidad definidos incorrectamente no estará disponible para los programas que necesitan acceder a él. En esta situación hipotética, el programa generalmente se interrumpirá y se encontrarán errores.