Traccia BW III

Lab- Navigating the Linux Filesystem and Permission Settings

Inizialmente, dopo aver creato un altro IDE Drive in UTM con dimensione di 1 GB, ho usato il comando lsblk per visualizzare tutti i dispositivi a blocchi: c'erano 2 blocchi sda e sdb, sda con una partizione sda1. Poi ho creato una partizione anche per sdb:

```
[analyst@secOps ~]$ lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda      8:0    0  10G  0 disk
└─sda1   8:1    0  10G  0 part /
sdb      8:16   0   1G  0 disk
[analyst@secOps ~]$ sudo fdisk /dev/sdb
[sudo] password for analyst:

Welcome to fdisk (util-linux 2.32).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x6643b349.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-2097151, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[analyst@secOps ~]$ sudo partprobe /dev/sdb
sudo: partprobe: command not found
[analyst@secOps ~]$ sudo partx -a /dev/sdb
partx: /dev/sdb: error adding partition 1
[analyst@secOps ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   10G  0 disk
└─sda1   8:1    0   10G  0 part /
sdb      8:16   0    1G  0 disk
└─sdb1   8:17   0 1023M  0 part
[analyst@secOps ~]$
```

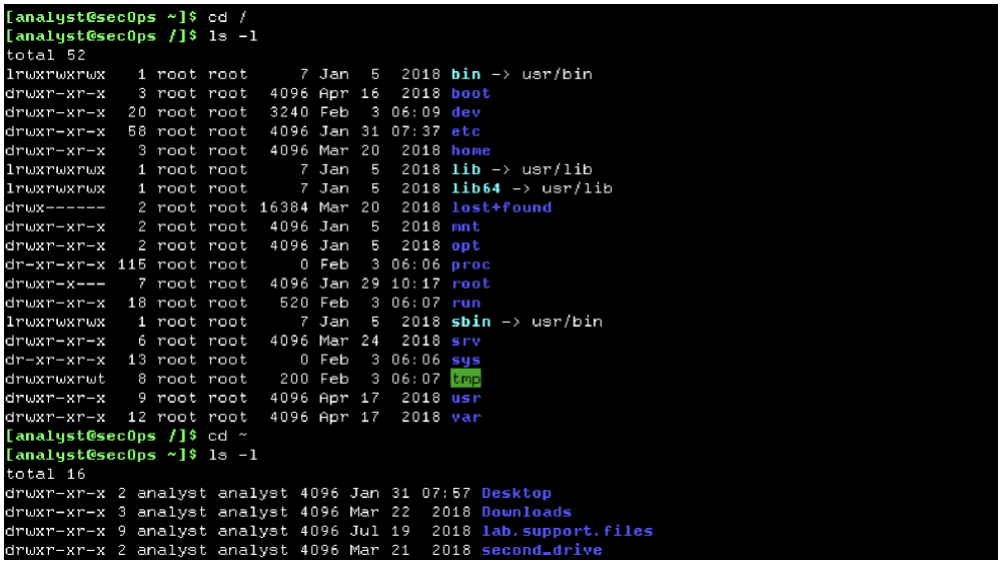
Il filesystem root memorizzato in /dev/sda1 è dove è memorizzato il sistema operativo Linux stesso. Di default tutti gli strumenti e i programmi sono memorizzati lì:

```
[analyst@secOps ~]$ mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
dev on /dev type devtmpfs (rw,nosuid,relatime,size=2016060k,nr_inodes=504015,mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
```

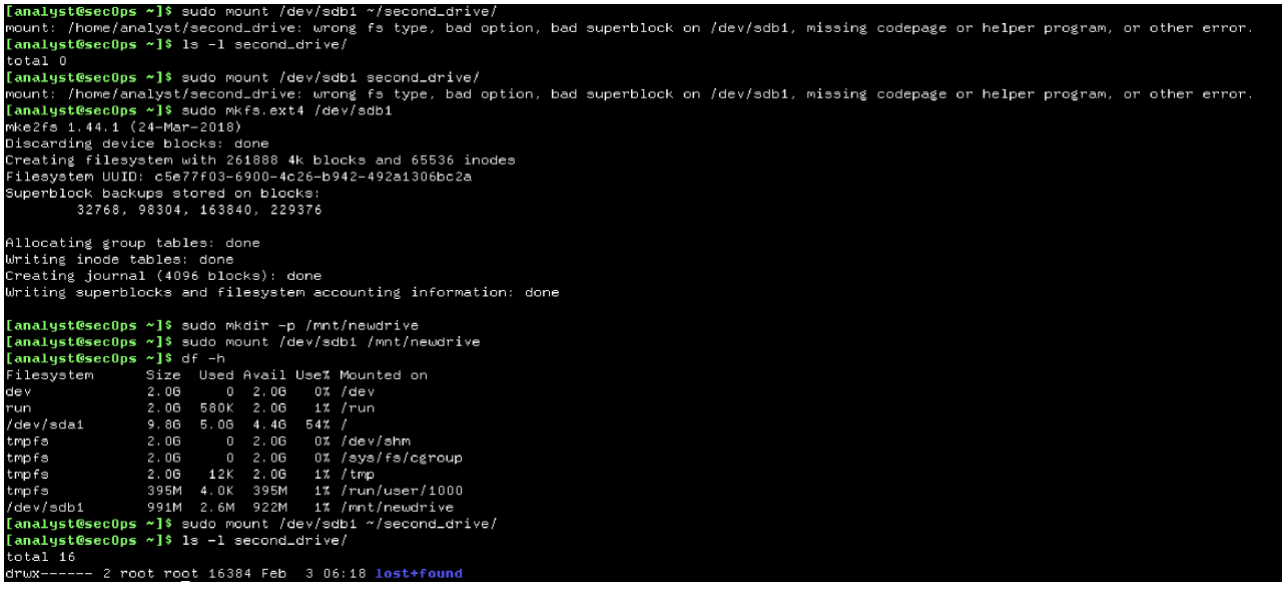
```
[analyst@secOps ~]$ mount | grep sda1
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
```

Come mostrato sopra, la radice si trova nella partizione di sda: sda1.

Quando andiamo nella directory root nel terminale e cerchiamo tutti i file, non vediamo dev/sdb1 lì. È perché non è attualmente montato:



Quindi, per montare manualmente il filesystem, possiamo usare il seguente comando sudo mount /dev/sdb1 ~/second_drive . Nel mio caso ho avuto un errore, quindi ho usato il comando sudo mkfs.ext4 /dev/sdb1 per risolverlo, che crea un nuovo filesystem Ext4 sulla partizione specificata e punta alla partizione in cui verrà creato il filesystem. Dopo di che, /dev/sdb1 viene montato in second_drive:



Ho usato di nuovo il comando mount per vedere informazioni dettagliate sulla partizione /dev/sdb1. Poi ho smontato la partizione:

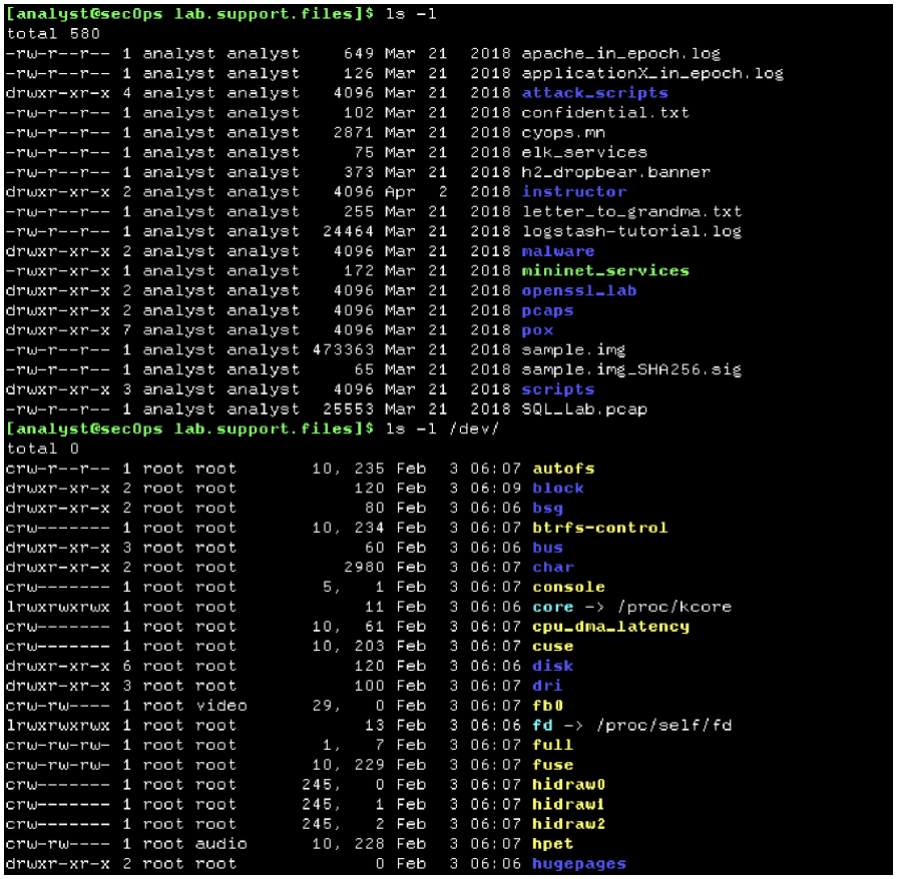
I filesystem Linux hanno funzionalità integrate per controllare la capacità degli utenti di visualizzare, modificare, navigare ed eseguire i contenuti del filesystem. Possiamo visualizzare e modificare i permessi. Ad esempio, se scriviamo ls -l lab.support.files/scripts/, possiamo vedere i permessi. Ad esempio, i permessi del proprietario e del gruppo di tutti i file e la directory in cui si trova analyst. A differenza di altri, il proprietario del file cyops.mn ha comandi di lettura e scrittura, ma non di esecuzione (-rw-r- -r- -). Per controllare i permessi della directory padre, ho provato a creare un file test1.tct usando il comando touch, tuttavia il permesso è stato negato. Questo perché la directory mnt ha i permessi per l'utente root. Quindi posso usare il comando sudo per creare il file sopra menzionato. Dopo aver creato il file, ho cambiato i permessi (chmod 665) e il proprietario (chown analyst) del file e ho aggiunto un testo al suo interno. Il 665 significa => il primo 6 è 4 + 2, il secondo 6 è 4 + 2, 5 è 4 + 1. Quindi, come si vede, il numero dopo chmod è di tre cifre. Il primo è per i Proprietari, il secondo è per il Gruppo e l'ultimo è per gli Altri. 4 significa Lettura (r), 2 significa Scrittura (w) e 1 significa Esecuzione (x). Dare permessi completi significa chmod 777 (4+2+1=7 per ciascuno):

```
[analyst@secOps scripts]$ ls -l
total 60
-rwxr-xr-x 1 analyst analyst  952 Mar 21  2018 configure_as_dhcp.sh
-rwxr-xr-x 1 analyst analyst 1153 Mar 21  2018 configure_as_static.sh
-rwxr-xr-x 1 analyst analyst 3459 Mar 21  2018 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 4062 Mar 21  2018 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 3669 Mar 21  2018 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Mar 21  2018 cyops.mn
-rwxr-xr-x 1 analyst analyst  458 Mar 21  2018 fw_rules
-rwxr-xr-x 1 analyst analyst   70 Mar 21  2018 mal_server_start.sh
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 net_configuration_files
-rwxr-xr-x 1 analyst analyst   65 Mar 21  2018 reg_server_start.sh
-rwxr-xr-x 1 analyst analyst  189 Mar 21  2018 start_ELK.sh
-rwxr-xr-x 1 analyst analyst   85 Mar 21  2018 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst   76 Mar 21  2018 start_pox.sh
-rwxr-xr-x 1 analyst analyst  106 Mar 21  2018 start_snort.sh
-rwxr-xr-x 1 analyst analyst   61 Mar 21  2018 start_tftpd.sh
[analyst@secOps scripts]$ touch /mnt/test1.txt
touch: cannot touch '/mnt/test1.txt': Permission denied
[analyst@secOps scripts]$ ls -ld /mnt/
drwxr-xr-x 3 root root 4096 Feb  3 06:19 /mnt/
[analyst@secOps scripts]$ sudo touch /mnt/test1.txt
[analyst@secOps scripts]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@secOps scripts]$ cd ~/second_drive/
[analyst@secOps second_drive]$ ls -l
total 16
drwx------ 2 root root 16384 Feb  3 06:18 lost+found
[analyst@secOps second_drive]$ sudo touch test2.txt
[analyst@secOps second_drive]$ ls -l
total 16
drwx------ 2 root root 16384 Feb  3 06:18 lost+found
-rw-r--r-- 1 root root     0 Feb  3 06:48 test2.txt
[analyst@secOps second_drive]$ sudo chmod 665 test2.txt
[sudo] password for analyst:
[analyst@secOps second_drive]$ sudo chown analyst test2.txt
[analyst@secOps second_drive]$ ls -l
total 16
drwx------ 2 root    root 16384 Feb  3 06:18 lost+found
-rw-rw-r-x 1 analyst root     0 Feb  3 06:48 test2.txt
[analyst@secOps second_drive]$ echo test >> test2.txt
[analyst@secOps second_drive]$ cat test2.txt
test
```

Finora abbiamo esplorato i diversi tipi di file in Linux e come sono rappresentati nell'output del comando ls -l. Il primo carattere nell'elenco indica il tipo di file. Ma in generale, ci sono tre categorie principali di file:

- File regolari (-)
- Questi includono file di testo, che sono leggibili dall'uomo; file binari, come programmi eseguibili; file di immagine; e file compressi utilizzati per l'ottimizzazione dell'archiviazione.
- File di directory (d)
- Questi rappresentano cartelle che organizzano file e altre directory.
- File speciali
- I file di blocco (b) sono utilizzati per interagire con l'hardware, come i dispositivi di archiviazione montati.
- I file di dispositivo a caratteri (c) gestiscono input e output seriali, come i terminali tty.
- I file di pipe (p) consentono ai dati di fluire in modalità FIFO (first-in, first-out) tra i processi.
- I file di collegamento simbolico (l) creano riferimenti ad altri file o directory, funzionando come scorciatoie.
- I file socket (s) facilitano la comunicazione tra applicazioni su una rete.

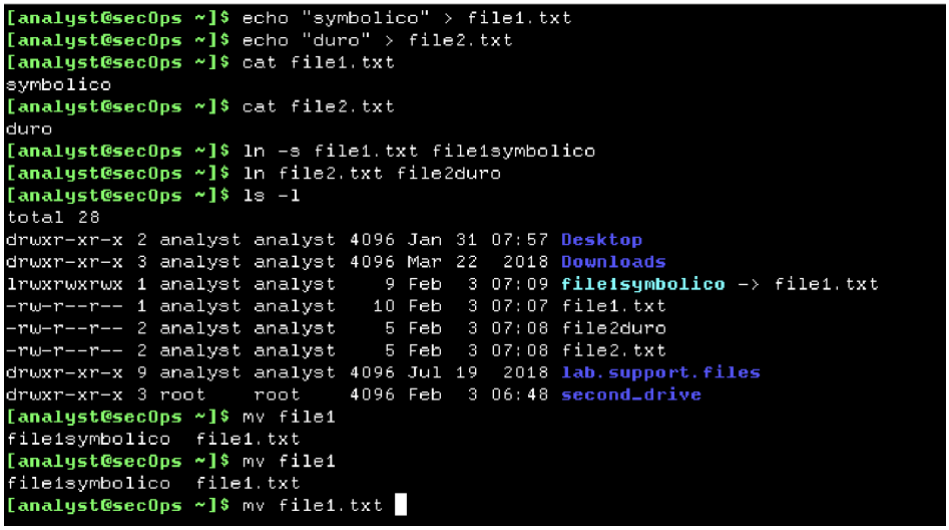
Li osserviamo anche nella seguente foto:



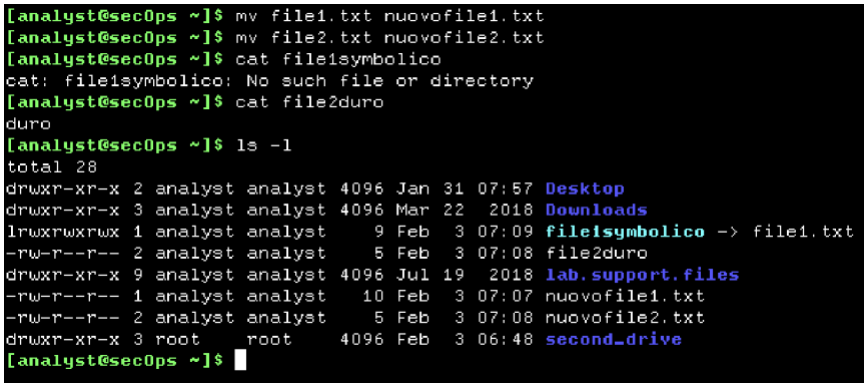
In Linux, i link simbolici funzionano come le scorciatoie in Windows. Esistono due tipi di link:

- Link simbolici (soft link) → Puntano al nome file di un altro file.
- Hard link → Puntano direttamente al contenuto di un altro file.

La differenza fondamentale è che se il file originale viene eliminato, un link simbolico si interrompe, mentre un hard link rimane intatto poiché fa riferimento agli stessi dati sul disco. Possiamo creare file usando echo e sperimentare con questi tipi di link. Ho usato ln –s per creare un link simbolico a file1.txt e ln per creare un hard link a file2.txt:



Ora cambiamo i nomi dei file originali: file1.txt e file2.txt, e osserviamo come influisce sui file collegati. Per questo ho usato il comando mv:



Abbiamo osservato come si comportano i collegamenti simbolici e fissi quando cambia il nome del file originale. Il collegamento simbolico file1symbolico si è interrotto perché puntava al nome file file1.txt, che è stato rinominato. Tuttavia, il collegamento fisso file2duro ha continuato a funzionare correttamente perché si collega direttamente all'inode di file2.txt, che è stato rinominato in nuovofile2.txt. Ciò ha confermato che i collegamenti simbolici dipendono dai nomi file, mentre i collegamenti fissi fanno riferimento alla struttura dati effettiva, rendendoli più resilienti alla ridenominazione.