Build Week 3 – Esercizio 3: Navigare nel Filesystem Linux e Impostazioni dei Permessi

Obiettivi:

- **Parte 1**: Esplorare i Filesystem in Linux
- **Parte 2**: Permessi dei File
- **Parte 3**: Link Simbolici e Altri Tipi di File Speciali

Risorse Richieste

- **CyberOps Workstation**

Parte 1: Esplorare i Filesystem in Linux

In questo laboratorio ci siamo concentrati sul filesystem ext, tra i più diffusi su Linux, anche se il sistema supporta molti altri tipi di filesystem. I filesystem devono essere montati prima di poter essere usati. Montare significa collegare la partizione fisica (hard disk, SSD, pen drive, ecc.) a una directory del sistema operativo. Questa directory, detta punto di montaggio, diventa la radice del filesystem e permette di accedere a tutti i file e le cartelle contenuti al suo interno.

Con il comando "*lsblk*" si ottiene l'elenco di tutti i dispositivi a blocchi presenti nel sistema.

```
[analyst@secOps ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
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
sr0     11:0    1 1024M  0 rom
```

Dall'output si vede che la VM ha tre dispositivi a blocchi: sr0, sda e sdb. I due hard disk, sda e sdb, contengono una sola partizione ciascuno e hanno dimensioni rispettivamente di 10 GB e 1 GB. In Linux i dischi sono rappresentati come /dev/sdX, con la lettera che distingue il disco e il numero finale che identifica la partizione.

Con il comando "*mount*" si possono visualizzare i filesystem attualmente montati nella VM, insieme ai relativi punti di montaggio.

```
[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=995444k,nr_inodes=248861,mode=755,inode64)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755,inode64)
/dev/sda1 on / type ext4 (rw,relatime)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot)
none on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=41,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=2096)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,nosuid,nodev,relatime,pagesize=2M)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev,nr_inodes=1048576,inode64)
tmpfs on /run/credentials/systemd-journald.service type tmpfs (ro,nosuid,nodev,noexec,relatime,nosymfollow,size=1024k,nr_inodes=1024,mode=700,inode64,noswap)
fusectl on /sys/fs/fuse/connections type fusectl (rw,nosuid,nodev,noexec,relatime)
configfs on /sys/kernel/config type configfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /run/credentials/systemd-networkd.service type tmpfs (ro,nosuid,nodev,noexec,relatime,nosymfollow,size=1024k,nr_inodes=1024,mode=700,inode64,noswap)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=200916k,nr_inodes=50229,mode=700,uid=1000,gid=1000,inode64)
```

Per il laboratorio ci concentriamo sul filesystem radice, memorizzato in "*/dev/sda1*", dove risiedono sistema operativo, programmi e file di configurazione. Con "*mount | grep sda1*" si filtra l'output per mostrare solo il filesystem radice.

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

L'output mostra che il filesystem radice è montato su "*/dev/sda1*", con punto di mount / e formattazione ext4; le parentesi indicano le opzioni di montaggio.

Con "*cd /*" ci si sposta nella root del filesystem, mentre "*ls -l*" elenca in dettaglio file e directory presenti nella radice.

```
[analyst@secOps /]$ ls -l
total 52
lrwxrwxrwx   1 root root     7 May  3 15:26 bin -> usr/bin
drwxr-xr-x   3 root root  4096 Jun 18 19:07 boot
drwxr-xr-x  20 root root  3920 Oct  3 04:44 dev
drwxr-xr-x  73 root root  4096 Jun 19 04:45 etc
drwxr-xr-x   3 root root  4096 Mar 20  2018 home
lrwxrwxrwx   1 root root     7 May  3 15:26 lib -> usr/lib
lrwxrwxrwx   1 root root     7 May  3 15:26 lib64 -> usr/lib
drwx------   2 root root 16384 Mar 20  2018 lost+found
drwxr-xr-x   2 root root  4096 Jan  5  2018 mnt
drwxr-xr-x   3 root root  4096 Jun 17 15:07 opt
dr-xr-xr-x 193 root root     0 Oct  3 04:44 proc
drwxr-x---   8 root root  4096 Jun 18 20:09 root
drwxr-xr-x  22 root root   580 Oct  3 04:44 run
lrwxrwxrwx   1 root root     7 May  3 15:26 sbin -> usr/bin
drwxr-xr-x   6 root root  4096 Mar 24  2018 srv
dr-xr-xr-x  13 root root     0 Oct  3 04:44 sys
drwxrwxrwt  11 root root   260 Oct  3 04:48 tmp
drwxr-xr-x  10 root root  4096 Jun 19 03:15 usr
drwxr-xr-x  12 root root  4096 Jun 19 04:45 var
```

Qual è il significato dell'output?
Mostra le directory principali presenti nella root del filesystem.

Dove sono fisicamente memorizzati i file elencati?
Si trovano fisicamente nella partizione "*/dev/sda1*".

Perché /dev/sdb1 non viene mostrato nell'output sopra?
Perché non è montata e quindi non è visibile nell'output.

Il comando "*mount*" serve anche a montare e smontare filesystem. Nella VM ci sono due dischi: "*/dev/sda*" e "*/dev/sdb*". Per montare un dispositivo è necessario prima creare un punto di montaggio. Con il comando "*ls -l*" si verifica che la cartella "*second_drive*" si trova nella home dell'utente analyst.

```
[analyst@secOps ~]$ ls -l second_drive/
total 0
```

La directory second_drive è inizialmente vuota. Con il comando "*sudo mount /dev/sdb1 ~/second_drive/*" si monta la partizione sdb1 su quella directory. Dopo il mount, con "*ls -l*" si vedono i file della partizione "*/dev/sdb1*" dentro la directory "*second_drive*".

```
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@secOps ~]$ ls -l
total 15496
drwxr-xr-x 2 analyst analyst     4096 Jun 17 19:35 Desktop
drwxr-xr-x 3 analyst analyst     4096 Jun 18 20:17 Downloads
-rw-r--r-- 1 root    root    14927893 Oct  1 04:01 httpdump.pcap
-rw-r--r-- 1 root    root      909925 Oct  1 04:16 httpsdump.pcap
drwxr-xr-x 9 analyst analyst     4096 Jun 18 20:17 lab.support.files
drwxr-xr-x 3 analyst analyst     4096 Jun 18 19:55 scripts
drwxr-xr-x 3 root    root        4096 Mar 26  2018 second_drive
drwxr-xr-x 5 analyst analyst     4096 Jun 18 19:27 yay
```

Perché la directory non è più vuota?
La directory "*second_drive*" non è più vuota perché è diventata il punto di montaggio della partizione "*/dev/sdb1*" e mostra i file presenti su quella partizione.

Dove sono fisicamente memorizzati i file elencati?
I file visibili dentro "*second_drive*" sono fisicamente memorizzati nella partizione "*/dev/sdb1*", non più nella partizione principale "*/dev/sda1*".

Con il comando "*mount*" si vedono le informazioni dettagliate su tutte le partizioni montate. Usando "*grep /dev/sd*" si filtrano solo i filesystem dei dischi, inclusa la partizione "*/dev/sdb1*".

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

Per smontare un filesystem basta uscire dal punto di montaggio e usare il comando umount sul dispositivo o sulla directory.

```
[analyst@secOps ~]$ sudo umount /dev/sdb1
[sudo] password for analyst:
[analyst@secOps ~]$ ls -l second_drive/
total 0
```

Parte 2: Permessi dei File

Ogni file in Linux ha un set di permessi che stabilisce cosa utenti e gruppi possono fare: leggere, scrivere o eseguire.

Con il comando "*cd /home/analyst/lab.support.files/scripts*" si entra nella directory indicata. Con "*ls -l*" si ottiene l'elenco dei file con i relativi permessi, proprietà e altre informazioni

```
[analyst@secOps ~]$ cd /home/analyst/lab.support.files/scripts
[analyst@secOps scripts]$ ls -l
total 68
-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 4053 Jun 18 20:09 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 5016 Jun 18 20:07 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 4189 Jun 18 19:22 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   86 Jun 18 20:27 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst   86 Jun 19 03:16 start_pox.sh
-rwxr-xr-x 1 analyst analyst  117 Jun 19 03:31 start_snort.sh
-rwxr-xr-x 1 analyst analyst   61 Mar 21  2018 start_tftpd.sh
```

Considera il file cyops.mn come esempio. Chi è il proprietario del file?
Il proprietario del file cyops.mn è l'utente analyst.

E il gruppo?
Il gruppo associato al file è anch'esso analyst.

I permessi per cyops.mn sono -rw-r-r-. Cosa significa?
I permessi -rw-r--r-- significano che:

- Il trattino iniziale indica che è un file regolare.
- rw- indica che il proprietario (analyst) può leggere e scrivere il file.
- r-- indica che il gruppo può solo leggere il file.
- r-- indica che tutti gli altri utenti possono solo leggere il file.

Il comando touch consente di creare velocemente un file vuoto, ad esempio "*touch /mnt/myNewFile.txt.*"

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

Perché il file non è stato creato? Elenca i permessi, la proprietà e il contenuto della directory /mnt e spiega cosa è successo. Con l'aggiunta dell'opzione -d, elenca i permessi della directory genitore. Registra la risposta nelle righe sottostanti.

Il file non è stato creato perché la directory /mnt appartiene a root e ha permessi drwxr-xr-x: solo root può scrivere al suo interno, mentre gli altri utenti hanno solo lettura ed esecuzione.

La directory /mnt ha:

- Permessi: drwxr-xr-x
- Proprietario: root
- Gruppo: root
- Contenuto: la directory risulta vuota

Con il comando "*ls -ld /mnt*" si conferma che i permessi della directory genitore permettono la scrittura soltanto a root.

```
[analyst@secOps scripts]$  ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan  5  2018 /mnt
```

Cosa si può fare affinché il comando touch mostrato sopra abbia successo?
l comando touch in /mnt può riuscire solo se l'utente ha permessi di scrittura. Le soluzioni possibili sono:

1. Eseguire il comando come root, ad esempio con sudo "*touch /mnt/myNewFile.txt*".
2. Cambiare la proprietà della directory /mnt all'utente analyst con sudo chown analyst:analyst /mnt.
3. Modificare i permessi della directory /mnt per consentire la scrittura a tutti, ad esempio con "*sudo chmod 777 /mnt*" (anche se questa opzione è sconsigliata per motivi di sicurezza).

Il comando "*chmod*" serve a modificare i permessi. Prima di usarlo, si monta di nuovo la partizione "*/dev/sdb1*" sulla directory "*/home/analyst/second_drive*" con "*mount /dev/sdb1 ~/second_drive*". Con "*cd second_drive*" si entra nella directory montata e con "*ls -l*" si elencano i file al suo interno.

```
[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 20
drwx------ 2 root    root    16384 Mar 26  2018 lost+found
-rw-r--r-- 1 analyst analyst   183 Mar 26  2018 myFile.txt
```

Quali sono i permessi del file myFile.txt?

I permessi iniziali del file myFile.txt sono **-rw-r--r--**:

- Proprietario: può leggere e scrivere.
- Gruppo: può solo leggere.
- Altri: possono solo leggere.

Con il comando "*chmod*" si modificano i permessi del file, ad esempio "*chmod 665 myFile.txt*".

```
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root    root    16384 Mar 26  2018 lost+found
-rw-rw-r-x 1 analyst analyst   183 Mar 26  2018 myFile.txt
```

I permessi sono cambiati? Quali sono i permessi di myFile.txt?

Sì, i permessi sono cambiati.

Dopo il comando "*chmod 665 myFile.txt*" i permessi risultano: -rw-rw-r-x

- Proprietario: lettura e scrittura
- Gruppo: lettura e scrittura
- Altri: lettura ed esecuzione

Usando il formato ottale, chmod 665 myFile.txt imposta i permessi così: 6 corrisponde a rw- (lettura e scrittura) per proprietario e gruppo, mentre 5 corrisponde a r-x (lettura ed esecuzione) per gli altri. In questo modo il file diventa leggibile e modificabile da proprietario e gruppo, e leggibile ed eseguibile da tutti gli altri utenti.

Quale comando cambierebbe i permessi di myFile.txt a rwxrwxrwx, garantendo a qualsiasi utente nel sistema pieno accesso al file?

Il comando è "*chmod 777 myFile.txt*".

Con il comando "*sudo chown analyst myFile.txt*" il file è stato reso di proprietà dell'utente analyst, ma appartiene ancora al gruppo root. .

```
[analyst@secOps second_drive]$  sudo chown analyst myFile.txt
[analyst@secOps second_drive]$ ls -l
\total 20
drwx------ 2 root    root    16384 Mar 26  2018 lost+found
-rw-rw-r-x 1 analyst analyst   183 Mar 26  2018 myFile.txt
```

Ora che analyst è proprietario del file, può modificarlo. Con il comando “*echo test >> myFile.txt*” aggiunge la parola “test” alla fine del file.

```
[analyst@secOps second_drive]$ echo test >> myFile.txt
[analyst@secOps second_drive]$  cat myFile.txt
This is a file stored in the /dev/sdb1 disk.
Notice that even though this file has been sitting in this disk for a while, it couldn't be accessed until the disk was properly mounted.
test
```

Sì, l’operazione è riuscita.
Essendo diventato proprietario del file, l’utente analyst dispone dei permessi di scrittura e può quindi accodare nuovo contenuto. Il comando “*echo test >> myFile.txt*” ha aggiunto correttamente la parola “test” alla fine del file.

Come i file, anche le directory hanno permessi. Con il comando “ls -l” in “/home/analyst/lab.support.files” si ottiene l’elenco dettagliato di file e directory con i rispettivi permessi.

```
[analyst@secOps second_drive]$  cd ~/lab.support.files/
[analyst@secOps lab.support.files]$ ls -l
total 580
-rw-r--r-- 1 analyst analyst    649 Mar 21  2018 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst    126 Mar 21  2018 applicationX_in_epoch.log
drwxr-xr-x 4 analyst analyst   4096 Mar 21  2018 attack_scripts
-rw-r--r-- 1 analyst analyst    102 Mar 21  2018 confidential.txt
-rw-r--r-- 1 analyst analyst   2871 Mar 21  2018 cyops.mn
-rw-r--r-- 1 analyst analyst     75 Mar 21  2018 elk_services
-rw-r--r-- 1 analyst analyst    373 Mar 21  2018 h2_dropbear.banner
drwxr-xr-x 2 analyst analyst   4096 Apr  2  2018 instructor
-rw-r--r-- 1 analyst analyst    255 Mar 21  2018 letter_to_grandma.txt
-rw-r--r-- 1 analyst analyst  24464 Mar 21  2018 logstash-tutorial.log
drwxr-xr-x 2 analyst analyst   4096 Mar 21  2018 malware
-rwxr-xr-x 1 analyst analyst    172 Mar 21  2018 mininet_services
drwxr-xr-x 2 analyst analyst   4096 Mar 21  2018 openssl_lab
drwxr-xr-x 2 analyst analyst   4096 Mar 21  2018 pcaps
drwxr-xr-x 6 analyst analyst   4096 Aug 15  2022 pox
-rw-r--r-- 1 analyst analyst 473363 Mar 21  2018 sample.img
-rw-r--r-- 1 analyst analyst     65 Mar 21  2018 sample.img_SHA256.sig
drwxr-xr-x 3 analyst analyst   4096 Jun 18 20:07 scripts
-rw-r--r-- 1 analyst analyst  25553 Mar 21  2018 SQL_Lab.pcap
```

Qual è la differenza tra la parte iniziale della riga di malware e la riga di mininet_services?

La differenza è nel tipo di oggetto indicato dal primo carattere della riga.

Per malware la riga inizia con "d", quindi è una directory con permessi rwxr-xr-x.
Per mininet_services la riga inizia con "-", quindi è un file regolare eseguibile con permessi rwxr-xr-x.

In sintesi, malware è una directory mentre mininet_services è un file.

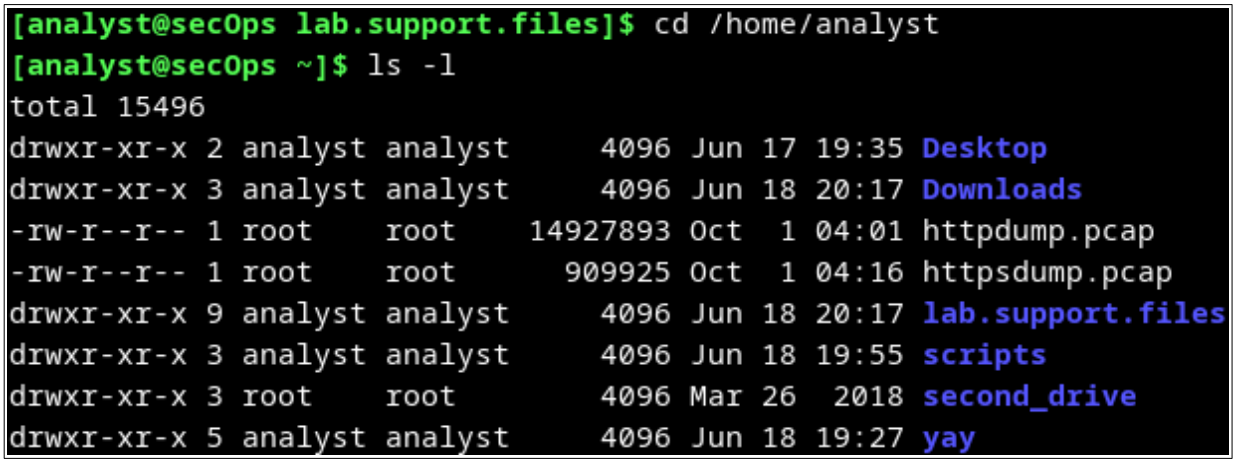
Parte 3: Link Simbolici e Altri Tipi di File Speciali

In Linux ogni file ha un tipo specifico che viene indicato dal primo carattere quando si usa il comando "*ls -l*". I tre tipi principali sono:

- File regolari, indicati con il simbolo "-". Questi possono essere di diversi sottotipi, ad esempio file di testo leggibili, file binari come programmi ed eseguibili, immagini o file compressi. Sono i file più comuni che l'utente utilizza quotidianamente.
- Directory, indicate dalla lettera "d". Sono cartelle che contengono altri file o sottodirectory e rappresentano la struttura gerarchica del filesystem.
- File speciali, che svolgono funzioni particolari nel sistema operativo. Tra questi troviamo:
  • File a blocco (b), usati per accedere a dispositivi hardware come gli hard disk.
  • File a carattere (c), che gestiscono flussi sequenziali di input e output, ad esempio i terminali tty.
  • File pipe (p), che permettono di passare informazioni tra processi in modalità FIFO (first in, first out).
  • Link simbolici (l), che collegano a un altro file o directory, con la possibilità di avere sia link simbolici che hard link.
  • Socket (s), utilizzati per lo scambio di dati tra applicazioni, ad esempio per la comunicazione su rete.

Questa distinzione permette a Linux di gestire in modo uniforme diversi tipi di risorse, trattando quasi tutto come se fosse un file.

Con "*ls -l*" nella cartella "*/home/analyst*" si vedono file e directory; il primo carattere indica il tipo: "-" per file e "d" per directory.



Con “*ls -l /dev*” si elencano i file di dispositivo. Nell’output si nota che i file a blocco iniziano con “b”, i file a carattere con “c” e i link simbolici con “l”.

```
[analyst@secOps ~]$  ls -l /dev/
total 0
crw-r--r--  1 root root       10, 235 Oct  3 04:44 autofs
drwxr-xr-x  2 root root           140 Oct  3 04:44 block
drwxr-xr-x  2 root root           100 Oct  3 04:44 bsg
crw-rw----  1 root disk       10, 234 Oct  3 04:44 btrfs-control
drwxr-xr-x  3 root root            60 Oct  3 04:44 bus
lrwxrwxrwx  1 root root             3 Oct  3 04:44 cdrom -> sr0
drwxr-xr-x  2 root root          3780 Oct  3 04:44 char
crw-------  1 root root        5,   1 Oct  3 04:44 console
lrwxrwxrwx  1 root root            11 Oct  3 04:44 core -> /proc/kcore
drwxr-xr-x  4 root root            80 Oct  3 04:44 cpu
crw-------  1 root root       10, 124 Oct  3 04:44 cpu_dma_latency
crw-------  1 root root       10, 203 Oct  3 04:44 cuse
drwxr-xr-x  7 root root           140 Oct  3 04:44 disk
drwxr-xr-x  2 root root            60 Oct  3 04:44 dma_heap
drwxr-xr-x  3 root root           100 Oct  3 04:44 dri
crw-rw----  1 root video      29,   0 Oct  3 04:44 fb0
lrwxrwxrwx  1 root root            13 Oct  3 04:44 fd -> /proc/self/fd
crw-rw-rw-  1 root root        1,   7 Oct  3 04:44 full
crw-rw-rw-  1 root root       10, 229 Oct  3 04:44 fuse
crw-------  1 root root      244,   0 Oct  3 04:44 hidraw0
crw-------  1 root root       10, 228 Oct  3 04:44 hpet
drwxr-xr-x  2 root root             0 Oct  3 04:44 hugepages
crw-------  1 root root       10, 183 Oct  3 04:44 hwrng
drwxr-xr-x  4 root root           360 Oct  3 04:44 input
crw-r--r--  1 root root        1,  11 Oct  3 04:44 kmsg
lrwxrwxrwx  1 root root            28 Oct  3 04:44 log -> /run/systemd/journal/dev-log
crw-rw----  1 root disk       10, 237 Oct  3 04:44 loop-control
drwxr-xr-x  2 root root            60 Oct  3 04:44 mapper
crw-r-----  1 root kmem        1,   1 Oct  3 04:44 mem
drwxrwxrwt  2 root root            40 Oct  3 04:44 mqueue
drwxr-xr-x  2 root root            60 Oct  3 04:44 net
crw-rw-rw-  1 root root        1,   3 Oct  3 04:44 null
crw-------  1 root root       10, 144 Oct  3 04:44 nvram
crw-r-----  1 root kmem        1,   4 Oct  3 04:44 port
crw-------  1 root root      108,   0 Oct  3 04:44 ppp
crw-------  1 root root       10,   1 Oct  3 04:44 psaux
crw-rw-rw-  1 root tty         5,   2 Oct  3 06:01 ptmx
```

```
brw-rw----  1 root disk        8,   0 Oct  3 04:44 sda
brw-rw----  1 root disk        8,   1 Oct  3 04:44 sda1
brw-rw----  1 root disk        8,  16 Oct  3 04:44 sdb
brw-rw----  1 root disk        8,  17 Oct  3 04:44 sdb1
drwxrwxrwt  2 root root           40 Oct  3 04:44 shm
crw-------  1 root root       10, 231 Oct  3 04:44 snapshot
drwxr-xr-x  3 root root          180 Oct  3 04:44 snd
brw-rw----+ 1 root optical    11,   0 Oct  3 04:44 sr0
```

In Linux esistono due tipi di collegamenti: i link simbolici, simili ai collegamenti di Windows, che puntano al nome di un file, e gli hard link, che puntano direttamente al contenuto del file. Per l'esercizio si inizia creando due file di prova con il comando "*touch*".

```
[analyst@secOps ~]$ touch file1.txt
[analyst@secOps ~]$ touch file2.txt
[analyst@secOps ~]$  echo "symbolic" > file1.txt
[analyst@secOps ~]$ cat file1.txt
symbolic
[analyst@secOps ~]$  echo "hard" > file2.txt
[analyst@secOps ~]$ cat file2.txt
hard
```

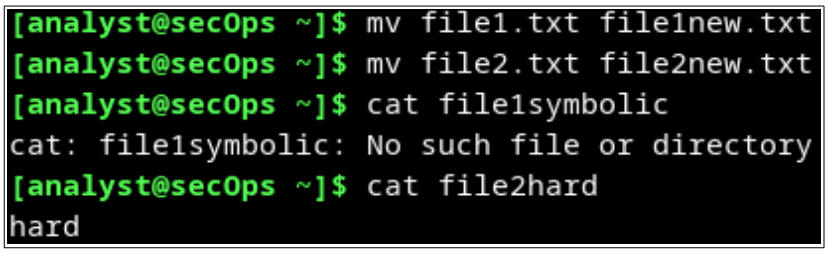
Con il comando ln -s file1.txt file1sym si crea un link simbolico a file1.txt. Con ln file2.txt file2hard si crea invece un hard link a file2.txt.

```
[analyst@secOps ~]$  ln -s file1.txt file1symbolic
[analyst@secOps ~]$ ln file2.txt file2hard
```

Con "*ls -l*" si può vedere l'elenco della directory e notare la presenza del link simbolico (indicato con "l") e dell'hard link, che invece appare come un normale file.

```
[analyst@secOps ~]$ ls -l
total 15508
drwxr-xr-x 2 analyst analyst     4096 Jun 17 19:35 Desktop
drwxr-xr-x 3 analyst analyst     4096 Jun 18 20:17 Downloads
lrwxrwxrwx 1 analyst analyst        9 Oct  3 06:07 file1symbolic -> file1.txt
-rw-r--r-- 1 analyst analyst        9 Oct  3 06:05 file1.txt
-rw-r--r-- 2 analyst analyst        5 Oct  3 06:06 file2hard
-rw-r--r-- 2 analyst analyst        5 Oct  3 06:06 file2.txt
-rw-r--r-- 1 root    root    14927893 Oct  1 04:01 httpdump.pcap
-rw-r--r-- 1 root    root      909925 Oct  1 04:16 httpsdump.pcap
drwxr-xr-x 9 analyst analyst     4096 Jun 18 20:17 lab.support.files
drwxr-xr-x 3 analyst analyst     4096 Jun 18 19:55 scripts
drwxr-xr-x 3 root    root        4096 Mar 26  2018 second_drive
drwxr-xr-x 5 analyst analyst     4096 Jun 18 19:27 yay
```

Rinominando “*file1.txt*”, il link simbolico file1symbolic si rompe perché punta al nome originale. Rinominando “*file2.txt*”, invece, file2hard resta valido perché è un hard link e continua a puntare allo stesso contenuto tramite “*inode*”.



Cosa pensi succederebbe a file2hard se aprissi un editor di testo e cambiassi il testo in file2new.txt?

Se modifichi il contenuto di “*file2new.txt*”, anche “*file2hard*” mostrerà le stesse modifiche. Questo accade perché “*file2hard*” è un hard link: non punta al nome del file, ma allo stesso inode su disco. Rinominare “*file2.txt*” in “*file2new.txt*” non cambia l’inode, quindi “*file2new.txt*” e “*file2hard*” restano collegati allo stesso contenuto fisico.