

Relazione: Navigazione nel Filesystem Linux e Impostazioni dei Permessi

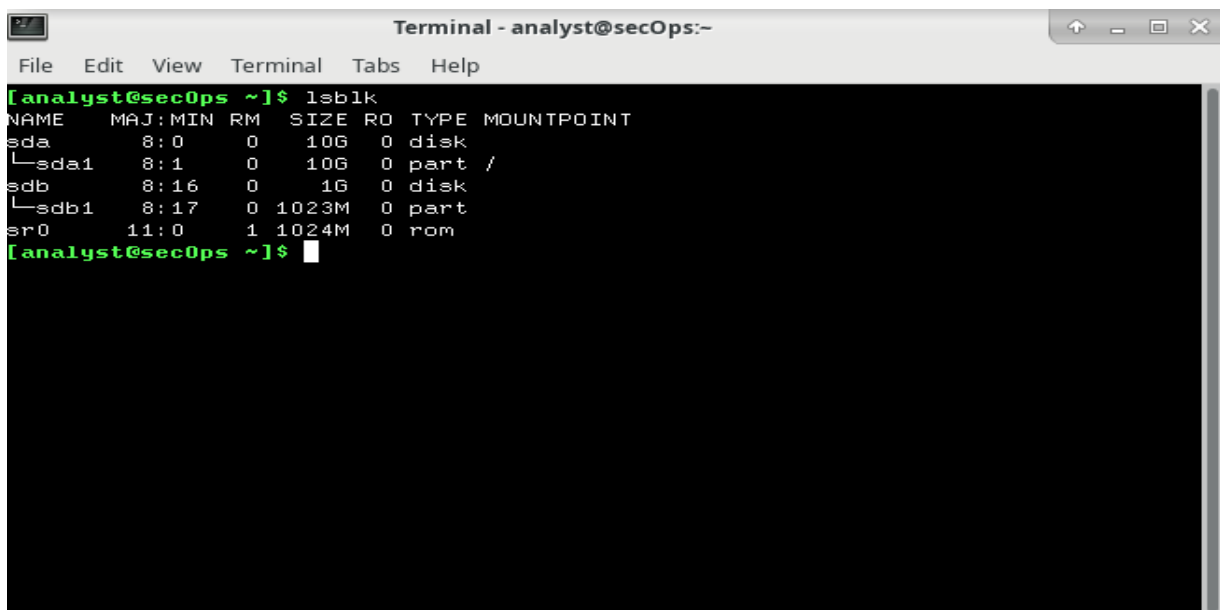
Introduzione

Questo laboratorio ha fornito un'esplorazione pratica del filesystem Linux e della gestione dei permessi, aspetti fondamentali per qualsiasi professionista della cybersecurity. L'obiettivo era familiarizzare con la struttura del filesystem, in particolare la famiglia ext, imparare a montare e smontare partizioni, comprendere e modificare i permessi di file e directory, e infine esplorare tipi di file speciali come i collegamenti simbolici. L'attività è stata svolta utilizzando la macchina virtuale CyberOps Workstation.

Parte 1: Esplorazione dei Filesystem in Linux

Il primo passo è stato accedere alla riga di comando della VM. Successivamente, abbiamo esplorato i filesystem montati. In Linux, un filesystem deve essere "montato" per essere accessibile. Montare significa collegare una partizione fisica (su un disco rigido, SSD, ecc.) a una directory specifica del sistema, chiamata punto di montaggio.

Abbiamo utilizzato il comando **lsblk** per visualizzare i dispositivi a blocchi disponibili:



```
Terminal - analyst@secOps:~  
File Edit View Terminal Tabs Help  
[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  
sr0         11:0    1 1024M  0 rom
```

Questo output mostra tre dispositivi: sda (un disco da 10GB), sdb (un disco da 1GB) e sr0 (probabilmente un'unità CD/DVD). Le partizioni sono indicate sotto i dischi (sda1, sdb1) [source: 10]. Linux usa la convenzione /dev/sdX per i dischi, dove X è una lettera, e un numero finale indica la partizione. Qui, /dev/sda1 è già montato sulla directory radice (/).

Per vedere i dettagli dei filesystem montati, abbiamo usato il comando mount. Poiché l'output era molto lungo, lo abbiamo filtrato con grep per isolare il filesystem radice:

```
[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=2015992k,nr_inodes=503998,mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nodelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore 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)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/bkio type cgroup (rw,nosuid,nodev,noexec,relatime,bkio)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=43,prgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=10856)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
configfs on /sys/kernel/config type configfs (rw,relatime)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=404332k,mode=700,uid=1000,gid=1000)
```

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

Questo conferma che /dev/sda1 è montato su / (il filesystem radice), utilizza il formato ext4 , e ha opzioni di montaggio come rw (lettura/scrittura) .

Successivamente, ci siamo spostati nella directory radice (cd /) e ne abbiamo elencato il contenuto (ls -l) :

```
[analyst@secOps ~]$ cd /
[analyst@secOps /]$ ls -l
total 52
lrwxrwxrwx   1 root root    7 Jan  5  2018 bin -> usr/bin
drwxr-xr-x   3 root root 4096 Apr 16  2018 boot
drwxr-xr-x  19 root root 3120 Apr 14 05:37 dev
drwxr-xr-x  58 root root 4096 Apr 17  2018 etc
drwxr-xr-x   3 root root 4096 Mar 20  2018 home
lrwxrwxrwx   1 root root    7 Jan  5  2018 lib -> usr/lib
lrwxrwxrwx   1 root root    7 Jan  5  2018 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   2 root root 4096 Jan  5  2018 opt
dr-xr-xr-x 134 root root    0 Apr 14 05:36 proc
drwxr-xr-x   7 root root 4096 Apr  9 09:13 root
drwxr-xr-x  17 root root  480 Apr 14 05:37 run
lrwxrwxrwx   1 root root    7 Jan  5  2018/sbin -> usr/bin
drwxr-xr-x   6 root root 4096 Mar 24  2018 srv
dr-xr-xr-x  13 root root    0 Apr 14 05:36 sys
drwxrwxrwt   8 root root  200 Apr 14 05:37 tmp
drwxr-xr-x   9 root root 4096 Apr 17  2018 usr
drwxr-xr-x  12 root root 4096 Apr 17  2018 var
```

Questi file e directory, essendo nella radice (/), sono fisicamente memorizzati sulla partizione /dev/sda1 . Il dispositivo /dev/sdb1 non appare qui perché non era ancora montato.

Abbiamo poi proceduto al montaggio manuale di /dev/sdb1. Per prima cosa, abbiamo verificato l'esistenza di una directory da usare come punto di montaggio nella home dell'utente analyst, chiamata second_drive :

```
[analyst@sec0ps ~]$ cd ~
[analyst@sec0ps ~]$ ls -l
total 24
-rw-r--r-- 1 root    root    5764 Apr  9 08:56 capture.pcap
drwxr-xr-x 2 analyst analyst 4096 Mar 22  2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22  2018 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 19  2018 lab.support.files
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 second_drive
```

Inizialmente, la directory era vuota:

```
[analyst@sec0ps ~]$ ls -l second_drive/
total 0
```

Abbiamo quindi montato la partizione /dev/sdb1 sulla directory ~/second_drive/

```
[analyst@sec0ps ~]$ ls -l second_drive/
total 0
[analyst@sec0ps ~]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
```

Dopo il montaggio, elencando nuovamente il contenuto della directory second_drive, abbiamo visto i file presenti sulla partizione /dev/sdb1

```
[analyst@sec0ps ~]$ ls -l second_drive/
total 20
drwx----- 2 root    root    16384 Mar 26  2018 lost+found
-rw-r--r-- 1 analyst analyst  183 Mar 26  2018 myFile.txt
```

La directory ~/second_drive è diventata il punto di accesso al filesystem memorizzato fisicamente su /dev/sdb1 [source: 45]. Verificando nuovamente con mount | grep /dev/sd, abbiamo visto entrambe le partizioni montate

```
[analyst@sec0ps ~]$ mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
/dev/sdb1 on /home/analyst/second_drive type ext4 (rw,relatime,data=ordered)
```

Infine, abbiamo smontato la partizione usando sudo umount /dev/sdb1 (assicurandoci di non essere all'interno della directory second_drive). Dopo lo smontaggio, la directory è tornata vuota:

```
[analyst@sec0ps ~]$ sudo umount /dev/sdb1
[analyst@sec0ps ~]$ ls -l second_drive/
total 0
```

Parte 2: Autorizzazioni dei File

Questa parte si è concentrata sui permessi dei file in Linux, che controllano l'accesso (lettura, scrittura, esecuzione) per proprietario, gruppo e altri utenti.

Ci siamo spostati in `/home/analyst/lab.support.files/scripts/` [source: 50] e abbiamo visualizzato i permessi con `ls -l`

```
[analyst@sec0ps ~]$ cd lab.support.files/scripts/
[analyst@sec0ps 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
```

Abbiamo analizzato il file `cyops.mn`. Il proprietario è `analyst`, il gruppo è `analyst`. I permessi `-rw-r--r--` significano:

- Proprietario (`analyst`): lettura e scrittura (`rw-`).
- Membri del gruppo (`analyst`): solo lettura (`r--`).
- Altri utenti: solo lettura (`r--`). Nessuno può eseguirlo.

Abbiamo tentato di creare un file vuoto in `/mnt` usando `touch`:

```
[analyst@sec0ps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
```

L'operazione è fallita a causa dei permessi. Abbiamo verificato i permessi della directory `/mnt` con `ls -ld /mnt`:

```
[analyst@sec0ps scripts]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan 5 2018 /mnt
```

La directory `/mnt` appartiene a `root` e i permessi (`drwxr-xr-x`) concedono il permesso di scrittura solo al proprietario (`root`). Per creare il file, avremmo dovuto usare `sudo touch` ... o modificare i permessi di `/mnt`.

Successivamente, abbiamo usato il comando `chmod` per modificare i permessi. Abbiamo rimontato `/dev/sdb1` su `~/second_drive`:

```
[analyst@secOps scripts]$ sudo mount /dev/sdb1 ~/second_drive/
[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
```

Ci siamo spostati in ~/second_drive e abbiamo controllato i permessi di myFile.txt
I permessi erano -rw-r--r-- . Li abbiamo modificati usando chmod in formato ottale

```
[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 in -rw-rw-r-x . Il valore ottale 665 si traduce in :

- Proprietario: 6 (ottale) = 110 (binario) = rw-
- Gruppo: 6 (ottale) = 110 (binario) = rw- [source: 73]
- Altri: 5 (ottale) = 101 (binario) = r-x [source: 73] Per dare permessi completi (rwxrwxrwx), il comando sarebbe sudo chmod 777 myFile.txt .

Abbiamo poi usato chown per cambiare il proprietario del file

```
[analyst@secOps second_drive]$ sudo chown analyst: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
```

Avendo (presumibilmente) cambiato il proprietario ad analyst e con i permessi -rw-rw-r-x ancora attivi, abbiamo provato ad aggiungere testo al 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
as properly mounted.
test
```

L'operazione è riuscita perché l'utente analyst è (o dovrebbe essere, secondo il comando chown eseguito) il proprietario e i permessi (rw- per il proprietario) consentono la scrittura .

Infine, abbiamo esaminato i permessi delle directory . Tornati in ~/lab.support.files/, abbiamo usato ls -l :

```

[analyst@sec0ps second_drive]$ cd ~/lab.support.files/
[analyst@sec0ps 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 7 analyst analyst 4096 Mar 21 2018 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 Mar 21 2018 scripts
-rw-r--r-- 1 analyst analyst 25553 Mar 21 2018 SQL_Lab.pcap

```

Confrontando la directory malware (drwxr-xr-x) con il file mininet_services (-rwxr-xr-x), la differenza principale è la lettera iniziale 'd' per la directory. Per le directory, il bit di esecuzione ('x') non significa "eseguibile" come per i file, ma "accessibile" o "attraversabile", cioè il permesso di entrare nella directory. I comandi chmod e chown funzionano sulle directory come sui file.

Parte 3: Collegamenti Simbolici e Altri Tipi di File Speciali

L'ultima parte ha esplorato diversi tipi di file in Linux, identificabili dal primo carattere nell'output di ls -l. Abbiamo visto:

- File regolari (-)
- Directory (d)
- File speciali :
 - Block device (b) (es. /dev/sda)
 - Character device (c) (es. terminali TTY, /dev/null)
 - Pipe (p) (FIFO)
 - Symbolic link (l)
 - Socket (s)

Abbiamo osservato file regolari e directory in /home/analyst :

```

-rw-r--r-- 1 analyst analyst 25553 Mar 21 2018 SQL_Lab.pcap
[analyst@secOps lab.support.files]$ cd ~
[analyst@secOps ~]$ ls -l
total 24
-rw-r--r-- 1 root root 5764 Apr 9 08:56 capture.pcap
drwxr-xr-x 2 analyst analyst 4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 19 2018 lab.support.files
drwxr-xr-x 3 root root 4096 Mar 26 2018 second_drive

```

E poi diversi tipi di file speciali nella directory /dev :

```

[analyst@secOps ~]$ ls -l /dev/
total 0
crw-r--r-- 1 root root 10, 235 Apr 14 05:37 autofs
drwxr-xr-x 2 root root 140 Apr 14 05:36 block
drwxr-xr-x 2 root root 100 Apr 14 05:36 bsg
crw----- 1 root root 10, 234 Apr 14 05:37 btrfs-control
drwxr-xr-x 3 root root 60 Apr 14 05:36 bus
lrwxrwxrwx 1 root root 3 Apr 14 05:37 cdrom -> sr0
drwxr-xr-x 2 root root 2800 Apr 14 05:37 char
crw----- 1 root root 5, 1 Apr 14 05:37 console
lrwxrwxrwx 1 root root 11 Apr 14 05:36 core -> /proc/kcore
crw----- 1 root root 10, 61 Apr 14 05:37 cpu_dma_latency
crw----- 1 root root 10, 203 Apr 14 05:37 cuse
drwxr-xr-x 6 root root 120 Apr 14 05:36 disk
drwxr-xr-x 3 root root 80 Apr 14 05:37 dri
crw-rw---- 1 root video 29, 0 Apr 14 05:37 fb0
lrwxrwxrwx 1 root root 13 Apr 14 05:36 fd -> /proc/self/fd
crw-rw-rw- 1 root root 1, 7 Apr 14 05:37 full
crw-rw-rw- 1 root root 10, 229 Apr 14 05:37 fuse
crw----- 1 root root 245, 0 Apr 14 05:37 hidraw0
crw-rw---- 1 root audio 10, 228 Apr 14 05:37 hpet
drwxr-xr-x 2 root root 0 Apr 14 05:37 hugepages
lrwxrwxrwx 1 root root 25 Apr 14 05:37 initctl -> /run/systemd/initctl/fifo
drwxr-xr-x 4 root root 360 Apr 14 05:37 input
crw-r--r-- 1 root root 1, 11 Apr 14 05:37 kmsg
drwxr-xr-x 2 root root 60 Apr 14 05:36 lightnvm
lrwxrwxrwx 1 root root 28 Apr 14 05:37 log -> /run/systemd/journal/dev-log
crw-rw---- 1 root disk 10, 237 Apr 14 05:37 loop-control
drwxr-xr-x 2 root root 60 Apr 14 05:37 mapper
crw-r----- 1 root kmem 1, 1 Apr 14 05:37 mem
crw----- 1 root root 10, 58 Apr 14 05:37 memory_bandwidth
drwxrwxrwt 2 root root 40 Apr 14 05:36 mqueue
drwxr-xr-x 2 root root 60 Apr 14 05:37 net
crw----- 1 root root 10, 60 Apr 14 05:37 network_latency
crw----- 1 root root 10, 59 Apr 14 05:37 network_throughput
crw-rw-rw- 1 root root 1, 3 Apr 14 05:37 null
crw-r----- 1 root kmem 1, 4 Apr 14 05:37 port
crw----- 1 root root 108, 0 Apr 14 05:37 ppp
crw----- 1 root root 10, 1 Apr 14 05:37 psaux
crw-rw-rw- 1 root tty 5, 2 Apr 14 06:02 ptmx
drwxr-xr-x 2 root root 0 Apr 14 05:36 pts
crw-rw-rw- 1 root root 1, 8 Apr 14 05:37 random
lrwxrwxrwx 1 root root 4 Apr 14 05:37 rtc -> rtc0
crw-rw---- 1 root audio 250, 0 Apr 14 05:37 rtc0
brw-rw---- 1 root disk 8, 0 Apr 14 05:37 sda
brw-rw---- 1 root disk 8, 1 Apr 14 05:37 sda1
brw-rw---- 1 root disk 8, 16 Apr 14 05:37 sdb
brw-rw---- 1 root disk 8, 17 Apr 14 05:37 sdb1
drwxrwxrwt 2 root root 40 Apr 14 05:36 shm
crw----- 1 root root 10, 231 Apr 14 05:37 snapshot
drwxr-xr-x 3 root root 180 Apr 14 05:37 snd
crw-rw----+ 1 root optical 11, 0 Apr 14 05:37 sr0
lrwxrwxrwx 1 root root 15 Apr 14 05:36 stderr -> /proc/self/fd/2

```

Ci siamo concentrati sui collegamenti (link). Esistono due tipi: simbolici (o soft link) e fisici (o hard link) . Un link simbolico punta al *nome* di un altro file, mentre un link fisico punta al *contenuto* (inode) di un altro file .

Abbiamo creato due file, file1.txt e file2.txt:

```
[analyst@sec0ps ~]$ echo "symbolic" > file1.txt
[analyst@sec0ps ~]$ cat file1.txt
symbolic
[analyst@sec0ps ~]$ echo "hard" > file2.txt
[analyst@sec0ps ~]$ cat file2.txt
hard
```

Poi abbiamo creato un link simbolico (ln -s) a file1.txt e un link fisico (ln) a file2.txt

```
[analyst@sec0ps ~]$ ln -s file1.txt file1symbolic
[analyst@sec0ps ~]$ ln file2.txt file2hard
```

L'output di ls -l ha mostrato la differenza:

```
[analyst@sec0ps ~]$ ls -l
total 36
-rw-r--r-- 1 root root 5764 Apr  9 08:56 capture.pcap
drwxr-xr-x 2 analyst analyst 4096 Mar 22 2018 Desktop
drwxr-xr-x 3 analyst analyst 4096 Mar 22 2018 Downloads
lrwxrwxrwx 1 analyst analyst  9 Apr 14 06:06 file1symbolic -> file1.txt
-rw-r--r-- 1 analyst analyst  9 Apr 14 06:03 file1.txt
-rw-r--r-- 2 analyst analyst  5 Apr 14 06:04 file2hard
-rw-r--r-- 2 analyst analyst  5 Apr 14 06:04 file2.txt
drwxr-xr-x 9 analyst analyst 4096 Jul 19 2018 lab.support.files
drwxr-xr-x 3 root root 4096 Mar 26 2018 second_drive
```

file1symbolic inizia con 'l', mostra la dimensione del percorso (9 byte) e punta (->) a file1.txt. file2hard appare come un file normale (-), ma ha un link count di 2 (quinta colonna), indicando che due nomi (file2.txt e file2hard) puntano allo stesso inode.

Abbiamo quindi rinominato i file originali file1.txt e file2.txt:

```
[analyst@sec0ps ~]$ mv file1.txt file1new.txt
[analyst@sec0ps ~]$ mv file2.txt file2new.txt
[analyst@sec0ps ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
[analyst@sec0ps ~]$ cat file2hard
hard
```

Tentando di leggere i link, abbiamo osservato il risultato

```
[analyst@sec0ps ~]$ mv file1.txt file1new.txt
[analyst@sec0ps ~]$ mv file2.txt file2new.txt
[analyst@sec0ps ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
[analyst@sec0ps ~]$ cat file2hard
hard
```

Il link simbolico file1symbolic si è rotto perché puntava al nome file1.txt, che non esiste più. Il link fisico file2hard funziona ancora perché punta all'inode, indipendentemente dal nome del file originale (ora file2new.txt). Se modificassimo il contenuto di file2new.txt, anche il contenuto letto tramite file2hard cambierebbe, poiché entrambi puntano agli stessi dati sul disco.

Riflessione

La comprensione dei filesystem, dei permessi e della proprietà dei file è cruciale in Linux . Errori nella configurazione dei permessi o della proprietà possono impedire ai programmi di accedere ai file necessari, causando malfunzionamenti ed errori . Questo laboratorio ha fornito una base solida per gestire questi aspetti fondamentali del sistema operativo.