appunti.md

LEZIONE 2, STORAGE MEDIA

Si parla dello storage locale. Lo storage è diviso in 3:

1. a blocchi
2. a file
3. a oggetti

Quello locale è a blocchi che possono essere partizionati con un *file system*. Partiamo dallo storage a blocchi per passare a quello a file. Si hanno diversi *file system* con diverse utilità, con diversi pro e contro. I più famosi sono ***ext4***, usata da *Ubuntu-Server* per esempio, ***XFS***, usata per esempio da *RHEL*, e ***BTRFS***, usata per esempio da *Suse*, che è una struttura dati complessa con volumi e partizioni logiche, tutto insieme al *file system*.

Uno *snapshot* è una sorta di "foto" di tutti gli storage, si congela lo stato attuale e si crea un file delle differenze sui cui andare avanti.

Per vedere i blocchi uso il comando *lsblk*:

```
[me@linuxbox ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  500G  0 disk
├─sda1   8:1    0    1G  0 part /boot
├─sda2   8:2    0  251G  0 part /home
├─sda3   8:3    0  240G  0 part /
├─sda4   8:4    0    1K  0 part
└─sda5   8:5    0    8G  0 part [SWAP]
sr0     11:0    1 1024M  0 rom
```

Si hanno due tipi di partition table, *GPT*, obbligatorio per dischi da più di 2TB, o *MSDOS*, che supporta 4 partizioni fisiche. All'inizio della partition table si scrivono l'inizio e la fine della partizione. Una volta creato un file system bisogna **montarlo**, ovvero scegliere un punto, un path nel directory tree, dove metterlo. Si può ovviamente smontare un disco. La rimozione sicura è il comando *sync* e il comando *umount*.

Vediamo l'uso di *fdisk* per manipolare le partizioni. Gli si da in pasto sempre il path di un disco, non della singola partizioni:

```
[me@linuxbox ~]$ sudo fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):

Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   g   create a new empty GPT partition table
   G   create an IRIX (SGI) partition table
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Con *n* creo una partizione che non si crea fino a *w* però. Si ha anche **parted** che crea subito e distrugge subito, NON CI SONO WRITE. Con *n* si segnala anche la famiglia del sile system. Il comando *partprobe* volendo forza la rilettura dei blocchi. Ora bisogna formattare la partizione con un file system, con *mkfs*, specificando il fs con "-t" e specificando poi la partizione:

```
[me@linuxbox ~]$ sudo mkfs -t ext4 /dev/sdb1
mke2fs 2.23.2 (12-Jul-2011)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
3904 inodes, 15608 blocks
780 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=15990784
2 block groups
8192 blocks per group, 8192 fragments per group
1952 inodes per group
Superblock backups stored on blocks:
  8193
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override
```

Ora bisogna montare con *mount*, specificando device e path:

```
[me@linuxbox ~]$ sudo mount /dev/sdb1 /mnt/flash
```

Se l'exit status, visibile con *echo $?* dopo l'ultimo comando, è 0 (e quindi visivamente non si hanno output) si ha che tutto è andato bene. Ci sono ovviamente dei path più adatti di altri, a causa delle routine del sistema... ma si può montare ovunque. Volendo *mount* mostra tutto ciò che è montato, ma anche anche non i blocchi quindi uso:

```
[me@linuxbox ~]$ mount | grep ^/
/dev/sda3 on / type ext4 (rw,relatime,seclabel,data=ordered)
/dev/sda1 on /boot type ext4 (rw,relatime,seclabel,data=ordered)
/dev/sda2 on /home type ext4 (rw,relatime,seclabel,data=ordered)
```

Infatti i blocchi iniziano per "/" mentre gli altri iniziano con i vari *cgroup* etc. Per smontare:

```
[me@linuxbox ~]$ sudo umount /dev/sdb1
```

o anche:

```
[me@linuxbox ~]$ sudo umount /mnt/flash
```

Il programma *lsof* di dice cosa sta usando un certo device:

```
[me@linuxbox ~]$ sudo lsof /home
bash      15106 linuxbox  cwd    DIR    8,2     4096 12320769 /home/linuxbox
...
```

Il file */etc/fstab* contiene le informazioni necessarie al montaggio delle periferiche di memorizzazione del sistema:

```
[me@linuxbox ~] cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Mon Feb 11 13:10:45 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=25e77052-3e2f-47a0-ba1d-4e3b5c698ac7 /                       ext4    defaults        1 1
```

```
UUID=ba4b75f6-ea8f-4ac4-a1ee-aae1f3198858 /boot                   ext4    defaults        1 2
UUID=f2a9fd86-def0-4c11-a57f-27d356d8905e /home                   ext4    defaults        1 2
UUID=6b075ea2-0273-4adc-8a80-68d58d55fe27 swap                    swap    defaults        0 0
```

Al posto dell'UUID specifico posso mettere anche il device nella forma */dev/...*.

Il comando *mount -a* monta quanto specificato nell'*fstab*. Con *df -h* vedo i path di quanto ho montato:

```
[me@linuxbox ~] df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       237G  4.2G  220G   2% /
devtmpfs        1.4G     0  1.4G   0% /dev
tmpfs           1.4G     0  1.4G   0% /dev/shm
tmpfs           1.4G  9.6M  1.4G   1% /run
tmpfs           1.4G     0  1.4G   0% /sys/fs/cgroup
/dev/sda1       976M  143M  767M  16% /boot
/dev/sda2       247G   67M  235G   1% /home
tmpfs           285M  4.0K  285M   1% /run/user/42
tmpfs           285M   40K  285M   1% /run/user/1000
tmpfs           285M     0  285M   0% /run/user/0
```

Se saturo un mount point lo rendo inutilizzabile. Un buon trucco è creare una cartella nel mountpoint e lavorarci dentro, così in uno script si può controllare l'esistenza della directory e abortire in caso di assenza.

LVM (APPROFONDIRE L'ARGOMENTO)

Vediamo ora come fare partizionamento dinamico, dove si usa una partizione o l'intero disco come fondamenta per una struttura dati di partizioni logiche. Si ha *LVM*, *Logical Volume Manager*, con lo storage vero definito da *Phisycal Volume* e un contenitore *Volume Group* dove credo dispositivi a blocchi logici, *Logical Volume*. Il kernel vede questi blocchi come fisici ma sono gestiti in maniera diversa. Espandere un volume logica significa rimappare l'*extend*. .

Con *fdisk* selezione **linux LVM**. Con *pvcreate* definisco il Phisycal Volume. *pvs* mostra tutti i volumi e *pvdispaly* seguito dal nome mostra le info riguardo uno in preciso. *vgcreate* crea il volume group e *vgs* mostra tutti i gruppi e *vgdisplay* seguito dal nome mostra le info riguardo uno in preciso. Poi ho *lvcreate* per creare il volume logico, con l'opzione "-L" per la dimensione, e "-n" per il nome, e con alla fine il nome del volume group. Si ha poi *lvdisplay* per le informazioni. A volte si ha il path esteso con */dev/mapper/...* Poi anche se si ha a che fare con volumi logici si usano comunque *mkfs, mount, ...* Si possono aggiungere ovviamente volumi logici all'fstab all'fstab. Gli UUID si vedono con *blkid*:

```
[me@linuxbox ~]$ sudo blkid /dev/sdb
/dev/sdb: PTUUID="4522fa09-fbe6-2144-a700-312b06b807df" PTTYPE="gpt"
```

Si ha che */boot* viene montato a parte perché grub al boot non ha caricato il modulo per LVM e quindi deve essere leggere i binari del kernel *linuz* in una partizione primaria.

ALTRO (DA RIGUARDARE)

Per fare un check su un file system uso *fsck*, da fare prima e dopo le manutenzioni, che a seconda del file system si fanno con file system montati o smontati:

```
[me@linuxbox ~]$ sudo fsck /dev/sdb1
fsck 1.40.8 (13-Mar-2016)
e2fsck 1.40.8 (13-Mar-2016)
/dev/sdb1: clean, 11/3904 files, 1661/15608 blocks
```

Abbiamo poi *dd*, device dump, che prende i blocchi da una sorgente e li mette da qualche altra parte, con la sintassi *dd if=input_file of=output_file [bs=block_size [count=blocks]]*. Prende in *of* anche file d'immagine. Il sorgente deve essere smontato. *dd* non ha controllo d'errore. Count se lasciato vuoto significa che copio tutto. Cancellando i primi 2048 byte per GPT (anche se ha il backup della partition table anche alla fine, e quindi va cancellata; per farlo si può usare *wipefs -a -f* per GPT) o 512 per MSDOS rendo irriconoscibile un device al kernel, infatti cancello la partition table, al più di consocere l'offset. Scrivo zero con dd:

```
[me@linuxbox ~]$ sudo dd if=/dev/zero of=/dev/sdb bs=512 count=1
```

CI sono dei file system speciali come *proc, sys etc...* e per questo si usa l'opzione *bind* di *mount*

Vediamo ora come montare dischi di rete, passiamo quindi allo storage a file. Si parla di *storage area network*. I più importanti protocolli di share di rete sono NFS e SMB, che però non è POSIX. Sul client devo avere installato *nfs-utils*. Poi ho la seguente sintassi, *nome_server:/cartella_share mount_point*:

```
[me@linuxbox ~]$ sudo mount server:/share /mnt/nfs/
```

Il controllo dell'utenza si fa lato server, controllando utente e gruppo... anche se non è il massimo in termini di sicurezza. Si usa per condividere file "a basso rischio". Ci sono soluzione più sicure. Per una share samba ho, dopo aver installato *cifs-utils*:

```
[me@linuxbox ~]$ sudo mount -o username=<user> //server/winshare /mnt/smb/
```

che chiede la password in quanto generalmente si avrà windows lato server con NTFS e non c'è il concetto di gruppo etc... Volendo si ha *samba-multiuser*, con un utente che fa il mount e tutti fanno il join.

Vediamo i permessi speciali, ovvero dei flag che impongono un certo comportamento. Sono 3:

1. setwid, vale 4, che applicato ad un eseguibile implica che lo si esegue con l'id proprietario del file, utile per /bin/passwd perché se no non tutti potrebbero scrivere su /etc/shadow. Viene ignorato sugli script (?)
2. setgid, vale 2 (?)
3. stickybit, vale 1, che dice che un utente può cancellare solo i suoi file (?)

infatti come gruppo ownr si ha il gruppo della share stessa, questo lo fa setgid, e si vede come "s" dando *ls -ld /share*.