
LINUX:
comandi principali,
mountpoint,
gestione utenti,
gestione processi

Principali comandi Linux

- **Per amministrare un sistema servono un certo numero di comandi**.
  - Nei sistemi Unix-Like (come Linux), per averne il manuale, è possibile usare il comando **`man`** seguito dal nome del comando in esame(es `man man` :)).
  - Spesso anche una ricerca in rete fornisce una versione web dello stesso manuale.
  - Risulta anche piuttosto utile il sito `http://explainshell.com/`, che tenta di dare una spiegazione, tratta dal manuale, dei comandi.
  - Quindi per ottenere aiuto sull'uso di un comando: `man <comando>`
    - Es: `# man ls`

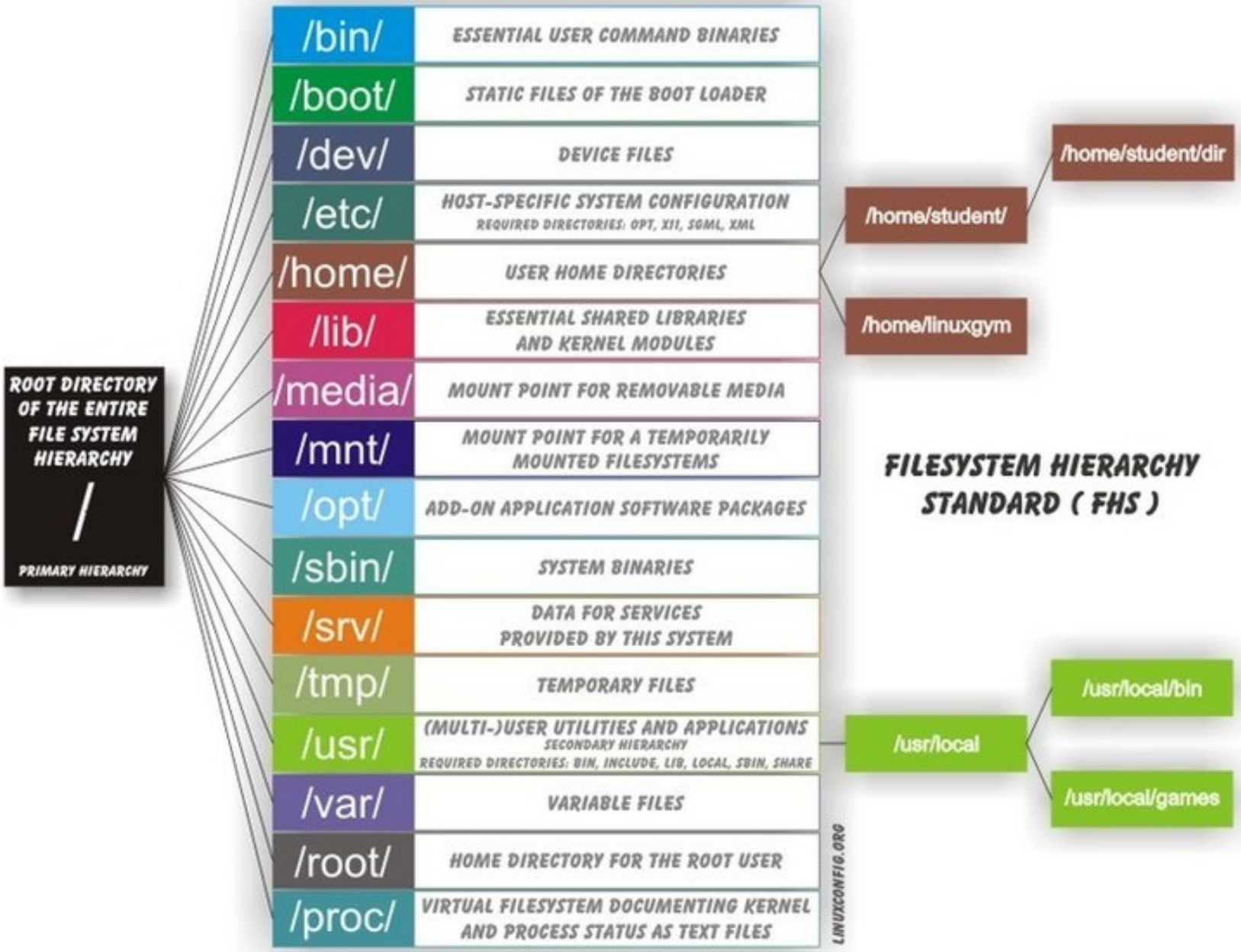
Principali comandi Linux

- In laboratorio useremo molti dei seguenti comandi:
  - Gestione dei file: `ls, cp, mv, rm, touch, head, tail, less, more, find`...
  - Gestione delle directory: `mkdir, rmdir, mv,cp.`
  - Gestione delle risorse: `df, du, free, top, htop.`
  - Spegnimento, riavvio: `shutdown, reboot, poweroff.`
  - Gestione **base** degli utenti: `adduser, addgroup, useradd, groupadd, usermod, chmod, chown, w, last,….`

Principali comandi Linux

- Gestione del software: `rpm, yast, urpmi, pacman, emerge, **dpkg**, **apt-get**, **apt-cache**, **apt**, snap.`
- Gestione dello spazio di memorizzazione di massa locale: `cfdisk, mkfs, mount, parted gparted, fdisk.`
- Configurazione della rete: `ifconfig, route, ip, netplan.`
- Controllo delle operazioni periodiche: `cron, at (/etc/crontab) e /etc/cron.d,...`

Struttura filesystem Linux



Struttura filesystem Linux

Mount

- Abbiamo visto che il filesystem di Linux è una **struttura gerarchica**, **complessa**, di **directory e sotto directory** cresciuta a dismisura negli anni.
- Inoltre, una unità di storage può essere suddivisa in varie **partizioni**. Ciò implica che ogni directory del filesystem di Linux possa occupare una partizione a se.
- Questo perchè **i sistemi ispirati ad UNIX hanno una singola gerarchia di memorizzazione, non esiste il concetto di lettera di unità come nei sistemi Windows**.
- Quindi come si estende il filesystem?

Mount - Mountpoint

- L'estensione della capacità del file system avviene tramite un'operazione detta **montaggio** (comando **`mount`**), in cui il file system di un'unità viene innestato in una directory del sistema detta **mountpoint**.
- La configurazione dei **mountpoint** (di sistema) è definita in **`/etc/fstab`.**
- Ad esempio la directory `/home`, con tutte le sue sotto-directory, potrebbe risiedere su un disco (o una partizione) a parte.
- Per verificare quando spazio disco è occupato sui vari file system montati, si può usare il comando **`df (-h)`**.
- Esempio di `fstab`:

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
/dev/sda1       /             ext4    errors=remount-     ro,….    0       1
/dev/sdc1    /home            ext4    defaults        0       2
/dev/sdb2       none             swap    sw              0       0
imagemaster.dsi.unive.it:/volume1/software /software nfs defaults   0       2
```

Mountpoint

- Esempio del comando `df`:

```
# df -h
Filesystem                              Size  Used Avail Use% Mounted
on
/dev/sda1                               459G   46G  390G  11% /
/dev/sdb1                               451G  306G  122G  72% /home
imagemaster:/volume1/software            11T  1,9T  9,0T  18%
/software
```

- Quando inserite una chiavetta USB in un sistema Linux, ad esempio Ubuntu, il sistema monta automaticamente il volume presente sulla chiavetta in una directory temporanea creata al volo avente come nome l'etichetta del volume della chiavetta nella posizione `/media/<username>/`:

```
# mount
/dev/sdd1 on /media/faromano/Fedora-WS-Live-25-1-3 type iso9660
(ro,nosuid,nodev,relatime,uid=1000,gid=1000,iocharset=utf8,mode=04
00,dmode=0500,uhelper=udisk
```

Filesystem in Linux/Unix - crearlo

- Per creare un file system si ricorre ai comandi della famiglia **`mkfs.*`** sul device file appropriato. Si noti che solitamente, quest'ultimo, non indica l'intero disco, ma piuttosto una partizione o, con una gestione più sofisticata, un volume logico.

```
# mkfs.ext4 /dev/sdd1

# mkfs.ntfs /dev/sda1
```

- Per creare partizioni su disco si possono usare, a seconda del tipo di partizionamento, **`[c]fdisk`** o **`[g]parted`**.
- La gestione dei volumi logici e dei sottostanti gruppi di volumi e volumi fisici tramite **LVM** è più complessa, e la loro trattazione richiederebbe troppo tempo, ma è generalmente consigliata, sopratutto in ambito server dove può capitare di dover ridimensionare al volo porzioni (**partizioni**) del disco.

Utenti e gruppi

- I sistemi simil-UNIX, tra cui Linux, sono generalmente multi-utente.
- Ogni utente è descritto dai seguenti campi:
  - un nome utente (**login name**),
  - un identificativo numerico (**user id o UID**),
  - un **gruppo primario** (vedremo poi a cosa serve),
  - una **home directory**,
  - una **login shell** ed
  - alcune informazioni aggiuntive, come ad es. il **nome reale**.
- **È importante notare che ciò che realmente identifica un utente è il suo UID**.
- I **gruppi**, come indica il nome, *raccolgono un insieme di utenti*, e vengono usati per assegnare dei permessi garantiti collettivamente a tutti i membri del gruppo.

Utenti e gruppi - comandi

- **whoami**: ottenere informazioni sul proprio username.
- **w**: ci dice che è collegato al sistema.
- **finger**: da informazioni dettagliate su un utente.

```
# finger romano

Login: romano                          Name: Fabrizio Romano

Directory: /home/romano                  Shell: /bin/bash

Last login Mon Jul 25 12:00 2016 (CEST) on pts/0 from 157.138.20.97

No mail.

No Plan.
```

- **id**: restituisce l'`uid` dell'utente e dei gruppi di cui fa parte:

```
# id faromano

uid=3033(faromano) gid=3000(Studenti)
groups=3000(Studenti),29(audio),24(cdrom),25(floppy),46(plugdev),44(video)
```

- Col comando **groups** è possibile vedere di quali gruppi fa parte un utente. Ad esempio:

```
# groups faromano

faromano : Studenti audio cdrom floppy plugdev video
```

Utenti e gruppi – utente root

- **In UNIX esiste un utente speciale, avente UID 0 e, generalmente, il nome root, che è sostanzialmente onnipotente, ed è usato dall'amministratore di sistema**.
  - Poiché da ***grandi poteri derivano grandi responsabilità*** (cit. Spiderman), l'amministratore usa un utente personale, e solo quando deve compiere operazioni che richiedano tali superpoteri, diventa **root** tramite il comando **`su`** o tramite **`sudo`**.
  - Mentre il comando **`su`** apre una shell eseguita come utente **root** chiedendo la password di quest'ultimo, il comando **`sudo`**, eventualmente seguito da un altro comando da eseguire come root, chiede la password dell'utente di partenza, a patto che questo sia in una lista/gruppo speciale (`admin, sudo, sudoers, ecc...`).
  - É possibile aggiungere un utente al gruppo `sudo` tramite il comando:

    ```
    # usermod -aG sudo <username>
    ```

- In Windows l'utente **root** trova il corrispondente in **Administrator**. (..e `sudo` diventa **`runas`**...)

Utenti e gruppi - permessi

- **Il modo in cui, in UNIX, si concede o si nega l'accesso a parti del filesystem a specifici utenti o gruppi è l'uso dei permessi**.
- Ogni oggetto del filesystem (file, directory) ha, infatti, associati un `utente` ed un `gruppo` che posseggono il file, nonché *almeno 9 bit di permessi*, più una *tripletta* aggiuntiva di *modificatori*. Si distinguono i permessi di lettura (**`r`**),scrittura (**`w`**) ed esecuzione (**`x`**) per ognuna delle classi utente proprietario (**`u`**), gruppo proprietario (**`g`**) e altri (**`o`**). I modificatori, sono, invece, `setuid, setgid e sticky bit`.

Utenti e gruppi - permessi

- Per cambiare il proprietario e il gruppo di un file si usano i comandi **chown** e **chgrp**, mentre per cambiare i permessi si usa il comando **chmod**.
- Ad esempio:

```
# chown faromano tmpfile.txt
# chown -R faromano:studenti tmpdir
# chgrp -R admin  tmpdir o chown :admin tmpdir

# chmod  ug+x pippo.txt
# chmod a+x  pippo.txt
# chmod o-x  pippo.txt
# chmod go+r  pippo.txt
# chmod u+rwx ,g+rw,o+r  pippo.txt
```

- u – user , g – group, o – others , a – all
- + assegna permessi , – rimuove permessi ,
- r w x read, write, execute, s sticky bit

Sticky Bit

- Al giorno d'oggi **lo sticky bit viene usato per le directory destinate a contenere file temporanei di più utenti**

- Se impostato su una directory esso **indica che i file e le subdirectory in essa contenuti possono essere cancellati o rinominat**i solo:
    - dal **proprietario**, o
    - dal **proprietario della directory** che lo contiene o
    - dall'utente **root**,

  anche se si dispongono di tutti gli altri permessi di scrittura necessari.

- Questo permesso viene spesso impostato sulle directory `/tmp` e `/var/tmp` per evitare che utenti ordinari cancellino o spostino i file temporanei appartenenti agli altri utenti, pur consentendo a chiunque di creare nuovi file e directory.

setuid

- Permette ad un utente che già possiede appropriati permessi di esecuzione su un determinato file di **eseguirlo con anche i privilegi dell'utente proprietario** del file oltre che ai propri.
- **Usato per permettere ad utenti non privilegiati di eseguire particolari programmi con i privilegi dell'amministratore** (`root`): un tipico esempio è l'assegnazione di tale permesso al comando `mount` per permettere anche agli utenti normali di montare dei file system residenti su memorie di massa rimovibili, quali chiavi USB o CD-ROM.

setgid

- Su file eseguibili si comporta in maniera analoga a `setuid`, con la differenza che il kernel attribuisce al nuovo processo un ***effective group ID*** pari a quello del gruppo assegnato al file, per cui il processo dispone anche dei privilegi di tale gruppo.
- Su file non eseguibili permette invece ai processi di usare su di essi il ***mandatory locking****:* un meccanismo di lock non aggirabile dai processi. Utile per alcuni file gestiti in `/tmp` e in `/run`.

Utenti e gruppi - permessi

- In tempi più recenti sono stati proposti molti miglioramenti a questo sistema di permessi, sia tramite l'uso di **attributi estesi** sia tramite **modelli di controllo alternativi della sicurezza** (ad esempio quelli basati su **ruoli**), che permettono una granularità molto più fine nel controllo dell'accesso ai file (e non solo). Tuttavia, ad oggi, non esiste un completo consenso su un modello da adottare, e pertanto **il minimo comune denominatore rimane sempre il sistema dei permessi**.
- NB: Windows utilizza un sistema di permessi molto più complesso basato su **ACL** o **ruoli** che possono essere applicate a gruppi o utenti singoli. Oltre all'interfaccia utente, per assegnare permessi si possono usare i comandi **`cacls`** e **`icacls`**.

Utenti e gruppi - Quota

- In sistemi condivisi è possibile assegnare ad ogni utente o gruppo di utenti una **quantità massima di spazio disco occupato** detta **QUOTA**.
- Vedremo più avanti come si configura, intanto accontentiamoci dei comandi per gestirla:

  `# quota nomeutente;` restituisce lo stato.

  `# edquota -u <utente>;` permette di cambiare la configurazione della quota di un utente.

  `# edquota -p <prototipo> <utente>;` assegna lo schema di quota dell'utente prototipo a utente.

Controllo dei processi

- **In UNIX ogni programma in esecuzione è rappresentato da almeno un processo**.
- I processi osservano una rappresentazione virtuale della memoria, che ne permette **l'isolamento**.
- Nel momento in cui viene lanciato, il processo assume (a meno dell'intervento dei modificatori `setuid` o `setgid` visti sopra) **un utente ed un gruppo proprietario** che sono quelli di chi ha eseguito il comando, e ne assume i diritti di accesso.
- Ogni processo è identificato da un **PID** (**process id**).

Controllo dei processi - comandi

- Per vedere la **lista dei processi attivi** si usa (di solito con vari parametri) il comando **`ps`** (`es: ps aux`).
- È inoltre possibile vedere la **genealogia dei processi** (chi ha lanciato chi) tramite il comando **`pstree`**.
- Una visione più dinamica dei processi in esecuzione si ha tramite il comando **`top`**, o la sua alternativa **`htop`**.
- È possibile mandare un **segnale** o **uccidere** un processo tramite il comando **`kill`** o **`killall`** seguito dal numero di segnale e dal **pid** (o **nome**) del processo:

```
# kill -9 2113

# killall -9 firefox

# kill $(pidof chrome)
```

- È inoltre possibile modificare la **priorità di esecuzione** di un processo usando i comandi:
  - `nice:` lancia un programma con una determinata priorità.
  - `renice:` cambia la priorità di un processo in esecuzione.
  - `ionice:` cambia la classe di I/O scheduling (*idle*, *realtime*, *best-effort*) e la priorità del processo.

Controllo dei processi

- Esempio di **`htop`**:

```
  1  [|||                                                                  2.6%]     Tasks: 65, 27 thr; 2 running
  2  [                                                                     0.0%]     Load average: 0.00 0.01 0.05
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 270/7985MB]     Uptime: 57 days, 03:28:31
  Swp[|                                                                0/4091MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
30393 root       20   0 18496  2848  1324 R  2.0  0.0  0:00.34 htop
 3277 casarin    20   0   98M  2208   884 S  0.7  0.0  1:57.20 sshd: casarin@pts/2
 3279 casarin    20   0  5416  1424   656 S  0.0  0.0  0:12.78 /usr/lib/openssh/sftp-server
30023 mfiorucci  20   0 14048  1432   956 S  0.0  0.0 16:06.43 watch  squeue -u mfiorucci
 1461 ntp        20   0 19656  1480   952 S  0.0  0.0  9:07.63 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 105:112
 1265 root       20   0  951M  1408   852 S  0.0  0.0  3:56.90 /usr/sbin/nscd
16892 root       20   0  951M  1408   852 S  0.0  0.0  0:06.48 /usr/sbin/nscd
29872 mfiorucci  20   0 18032  3128   792 S  0.0  0.0 17:42.66 tmux
 1288 root       20   0  951M  1408   852 S  0.0  0.0  0:06.50 /usr/sbin/nscd
    1 root       20   0 35616  2680  1280 S  0.0  0.0  0:04.57 /sbin/init
  329 root       20   0 19484   640   444 S  0.0  0.0  0:00.56 upstart-udev-bridge --daemon
  335 root       20   0 51528  1356   792 S  0.0  0.0  0:01.13 /lib/systemd/systemd-udevd --daemon
  503 root       20   0 15268   632   396 S  0.0  0.0  0:00.19 upstart-socket-bridge --daemon
  683 root       20   0 23428   968   664 S  0.0  0.0  0:11.58 rpcbind
  758 statd      20   0 21548  1160   704 S  0.0  0.0  0:00.00 rpc.statd -L
  953 root       20   0 15284   404   192 S  0.0  0.0  0:00.14 upstart-file-bridge --daemon
 1008 syslog     20   0  251M  1460   892 S  0.0  0.0  0:01.51 rsyslogd
 1009 syslog     20   0  251M  1460   892 S  0.0  0.0  0:00.00 rsyslogd
 1010 syslog     20   0  251M  1460   892 S  0.0  0.0  0:02.39 rsyslogd
 1005 syslog     20   0  251M  1460   892 S  0.0  0.0  0:03.95 rsyslogd
 1007 root       20   0 23484   360   160 S  0.0  0.0  0:00.52 rpc.idmapd
 1407 root       20   0 19920  1524  1244 S  0.0  0.0  0:19.92 /usr/lib/postfix/master
 1417 postfix    20   0 21716  1612  1316 S  0.0  0.0  0:06.58 qmgr -l -t unix -u
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Altre informazioni sul sistema

- È possibile ottenere altre informazioni sullo stato del sistema, oltre che dai già visti comandi `df`, `[h]top` e `cat /proc/cpuinfo`, dai comandi **`free`** (uso della memoria), **`iotop`** (visione dinamica dell'uso dei sistemi di input/output), **`vmstat`** (informazioni varie su memoria virtuale e i/o), **`dmesg`** (ultimi messaggi del kernel), **`lspci`**, **`lsusb`**, ...

- Un comando molto utile è **`lsof`**, **che mostra i file aperti dai processi**, rendendo possibile identificare quali programmi stanno, ad esempio, bloccando l'operazione di smontaggio di un filesystem.

Gestione della rete

- È possibile vedere le connessioni di rete (o tramite unix socket) effettuate dai processi in esecuzione, nonché altre informazioni, tramite il comando **`netstat.`** Ad esempio lanciate**:**

```
# netstat --inet -an
```

  - Dove **`--inet`** indica solo connessioni IP, **`a`** tutti i socket, **`n`** mostra solo gli ip degli host collegati.

- la gestione delle interfacce avviene tramite il comando **`ifconfig (o ip)`** e quella delle tabelle di instradamento tramite il comando **`route(o ip):`**

```
# ifconfig eth0 157.138.22.21 netmask 255.255.255.0
up
```

  - Abilita la scheda di rete **`eth0`** assegnado l'ip **`157.138.22.21`** con netmask **`255.255.255.0.`**

```
# route add default gateway 157.138.22.1
```

  - Imposta il gateway di default a **`157.138.22.1`**

```
# ip a add 192.168.100.200/24 dev eth0
# ip route add 192.168.100.0/24 via 192.168.100.1
```

Gestione della rete

- Nei sistemi Debian/Ubuntu il file principale per la configurazione della rete ora è `/etc/netplan/*.yaml`.
- In precendenza file file di configuazione di rete era **`/etc/network/interfaces`**, che veniva letto dai comandi **`ifup`** e **`ifdown`**, i quali attivano e disattivano le varie schede di rete come specificato dal file. Debian usa ancora questo sistema.
- Per quanto riguarda la configurazione dei **Domain Name Server** (**DNS**) da consultare, ci si riferisce al file **`/etc/resolv.conf`**,ora generato dinamicamente a partire dalle configurazioni specificate in `(/etc/netplan/*.yaml )`.
- **`traceroute`**: stampa il percorso(**route**) che fanno i pacchetti dall'host su cui è lanciato fino all'host di destinazione.

```
# traceroute www.google.it
```

In Windows

- La maggior parte delle operazioni si fa tramite **interfaccia grafica**. In particolare:
  - **Active Directory Users and Computers**: per la gestione del catalogo utenti.
  - **DNS**: per la gestione di un server dns.
  - **Share**: per la condivisione windows di file e cartelle.
  - **Event viewer**: per la consultazione dei log di sistema.
  - **File explorer**: utile per la gestione dei permessi.

- Ci sono però anche dei comandi manuali….

In Windows… comandi manuali

- Gestione della rete: `ipconfig, netsh, ping, tracert`

```
C:\> netsh -c interface dump > rete.txt
C:\> netsh -f rete rete.txt
```

- Gestione filesystem: `mount, copy, del, move….`

```
C:\> NET USE X: \\imagemaster.dsi.unive.it\Software
/PERSISTENT:NO
```

- Gestione degli utenti: `dsadd, dsrm,….`

```
C:\> dsadd user
CN=John,CN=Users,DC=it,DC=uk,DC=savilltech,DC=
com -samid John -pwd Pa55word
```