

# Altre nozioni di Linux e comandi su bash

Il programma che ci permette di interpretare i comandi per poi mandarli al sistema operativo si chiama **Shell**.

Quindi nasce come **ambiente di scripting**, cioè sequenze di istruzioni che servono a risolvere un problema.

## BASH

E' la shell di default sta per Bourne-again Shell, scritta nel 1977 da Stephen Bourne.

Linux si basa sull'approccio **divide et impera** ogni comando esegue un compito atomico e lo svolge al meglio, possiamo **applicare i comandi su file o sequenza di byte**, nasce per elaborare file e come sappiamo tutto è un file.

I comandi basilari di bash si fondano sull'idea di:

- Apertura
- Scrittura
- Lettura
- Ricerca

In un file.

## Particolarità:

I buffer di questi comandi sono limitati per fare operazioni semplici ed evitare di sovraccaricare il processore.

**Standard Output:** Si riferisce allo **stream standardizzato dei dati prodotti dalla linea di comando in Linux/Unix.**

I vari processi sulla shell non condividono lo spazio di memoria per evitare che un processo possa modificarne un altro.

Esistono inoltre uno strumento che serve a far comunicare i processi: la **PIPE**.

## Perché Linux è così veloce?

Le istruzioni di Linux sono scritte in C, un linguaggio molto veloce, inoltre sono state scritte appositamente per le macchine su cui vengono girate.

Uno dei pochi comandi che ha bisogno di leggere tutto il file, il che rallenta la velocità d'esecuzione, è il **Sort**, perché per poter ordinare  $n$  righe le dovrà necessariamente vedere tutte.

## Variabili dell'ambiente shell

Le variabili sono **locazioni di memoria nominate**, le variabili di ambiente sono usate dalla shell per immagazzinare delle **informazioni**. Le variabili d'ambiente sono **condivise tra i programmi che la shell esegue**.

### PATH

Il path è una variabile che serve a far capire alla shell dove trovare ed eseguire determinati comandi. concatenazione di percorsi con cui cerchiamo le directory che ci interessano.

Esempio:

```
/home/francesco/Downloads/file
```

Sono variabili di ambiente perché specificano a tutti i programmi i valori in quella shell che sta venendo eseguita.

### Esempi di variabili e comandi da usare su variabili:

```
echo $PATH //Mostra il path corrente
printenv //Mostra tutte le variabili di ambiente attive
foo = 5 //Imposta la variabile foo a 5
whoami //Nome utente
hostname //Nome del computer
date //Data e tempo attuale
cal //Calendario
echo my login is $(whoami)
$(comando) //Leggi lo standard output del comando e usalo come una
variabile
```

Se vogliamo vedere vecchi comandi usati basta usare **history**.

⚠ Ogni variabile d'ambiente muore nel momento in cui chiudo la shell.

## Differenza tra help e man

Il comando **help** si usa quando dobbiamo andare a cercare il comando che vogliamo utilizzare oppure vedere delle specifiche del comando.

**man** è invece il manuale effettivo del comando, quindi per utilizzarlo dobbiamo già essere a conoscenza dell'esistenza del comando. Solitamente **man** è usato con **less**.

✍ I comandi utilizzano altri comandi elementari per dare un supporto.

## Redicter

Ogni programma sputa sullo standard output i byte che produce possiamo usare la **pipe** (**|**) per concatenare più comandi, con la quale possiamo fare operazioni in cascata.

Possiamo quindi concatenare comandi che fanno operazioni atomiche in comandi più complessi:

```
w | grep 'user' | sed s/tuta/scholar/g
```

Possiamo inoltre scrivere i risultati di un comando su un file invece che sullo standard output con il simbolo '>':

```
w | awk -F" " '{print $1}' | sort | uniq > users
```

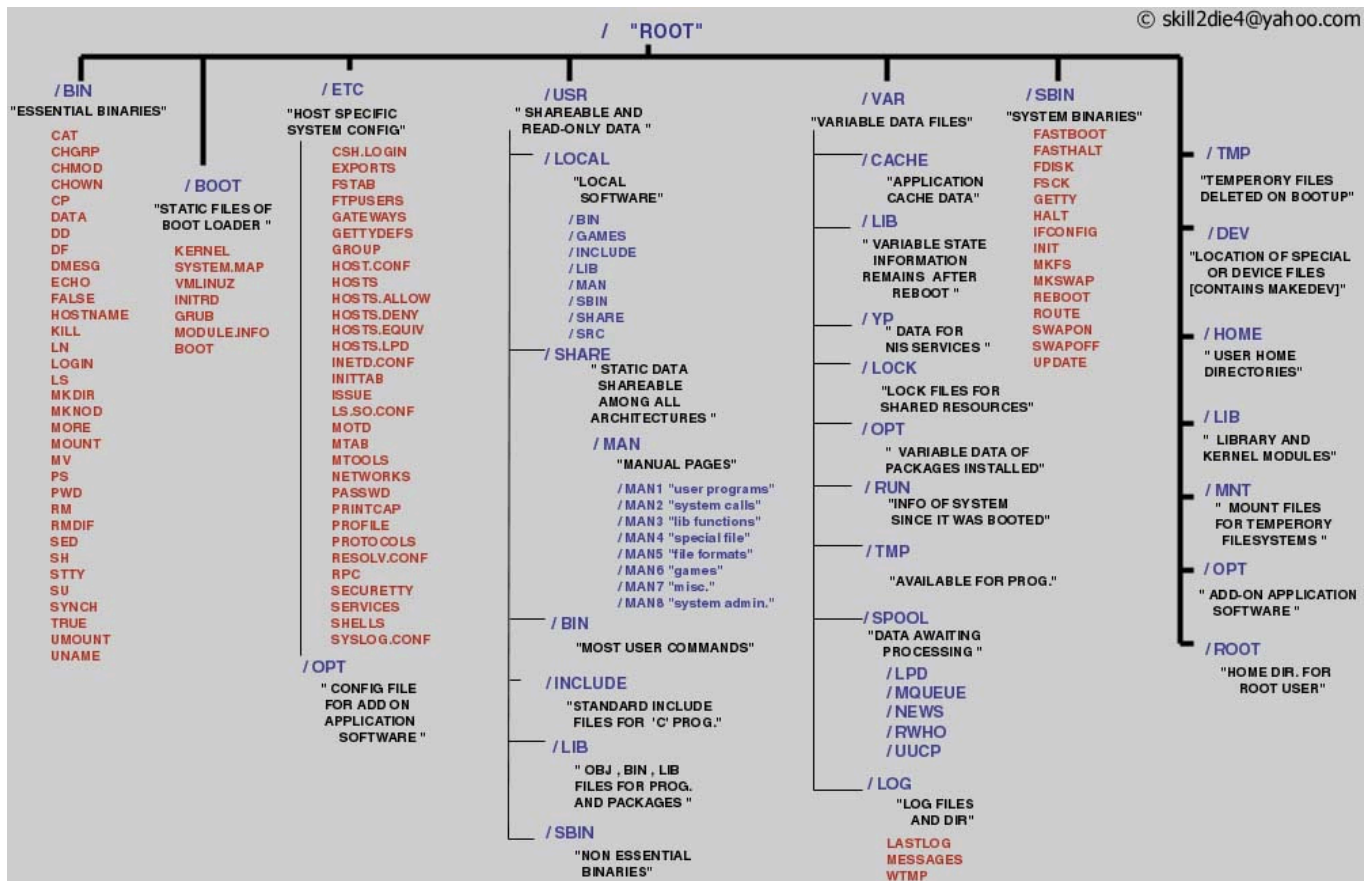
L'estensioni dei file su Linux non esistono poiché è una comodità dell'utente e non del SO, l'utente si deve prendere la responsabilità di conoscere in anticipo i file che usa.

✍ Solitamente se un file non ha un'estensione allora è un programma.

## File system in Linux

La struttura del file system è simile ad un albero al contrario. Ci sta una rigida convenzione di struttura delle cartelle

Nasce tutto nella cartella root, le cartelle di primo livello sono accedibili soltanto dall'amministratore e l'unica cartella di primo livello accessibile a tutti è **TMP**.




Ad ogni utente viene creata una cartella per l'utente nella directory home.

 **L'utente è proprietario della sua cartella e non della home.**

Per prassi quando l'utente viene creato viene anche creato anche un **gruppo di appartenenze** dove ci sta solamente lui. Il nome dell'utente solitamente non è il nome del gruppo.

I comandi principali per muoversi nel file system sono:

- **pwd**: print current directory;
- **ls**: list files;
- **cd**: change directory

 **Differenza tra execute e read nelle cartelle di Linux:**

- **Read**: Ci permette di vedere i file all'interno della cartella;
- **Execute**: Ci permette di entrare nella cartella ed accedere ai file.

**Alcune cartelle da ricordare:**

- **/BIN**: Contiene tutti gli ==essential binaries==(gestione minimale);

- **/BOOT**: informazioni necessarie per accendere il computer;
- **/ETC**: contiene tutti i file di configurazione della macchina;
- **DEV**: contiene tutte i file astratti come stampanti;
- **/LIB**: contiene le librerie;
- **/MNT**: le pennette vengono aggiunte come una cartella qui:
- **/ROOT**: generalmente mess fuori dalla home;
- **/OPT**
- **/TMP**: file temporanei;
- **/USR**: informazioni a disposizione di tutti gli utenti, ma in sola lettura;
- **/VAR**: contiene variabili utili per le applicazioni quindi contiene tutto quello che viene gestito e mantenuto nel tempo (ex: log, cache. YP, lock .....);

Il pathname può essere relativo o assoluto per muoverci nelle directory.

Esistono anche dei caratteri particolari:

- **tilde**: Ti fa tornare sulla home;
- **'.'**: cartella in cui mi trovo ;
- **'..'**: la cartella superiore;
- **'\*'**: wildcard matching any filename;
- **'?'**: wildcard matching any charcter;
- **'TAB'**: Completa il nome del file se esiste un nome univoco;

Possiamo usare **ls** con diversi parametri anche concatenati:

Esempi:

- **ls -l**;
- **ls -lt**;
- **ls -lr**;
- **ls -f**.
- Molti altri.

Un file nascosto in una cartella contiene il punto davanti nel nome e non sono necessariamente file da tenere nascosti.

**mv** crea una copia da un'altra parte e poi elimina quella origininale(copia e cancella).

## Permessi di accesso

Gli accessi ai file sono descritti da una sequenza di tre terne di bit che vanno a specificare quali operazioni possono svolgere: utente, gruppo, other.

Per cambiare i permessi di accesso utilizziamo il comando **chmod**:

```
chmod ug+x foo
```

```
//Dai il permesso di esecuzione (x) ad utente (u) e gruppo(g)
```

```
//Ovviamente ci sono anche o(other), r(read) e w(write)
```

Esiste anche una mappatura numerica di questi permessi d'esecuzione:


#	Permission	rwX
0	none	000
1	execute only	001
2	write only	010
3	write and execute	011
4	read only	100
5	read and execute	101
6	read and write	110
7	read, write, and execute	111

## Gestione dei processi

Ogni processo quando ne esegue un altro crea un processo figlio a cui viene dato un **PID**, se volessi vedere le informazioni del processo ho bisogno del **PID** per farlo utilizzo il comando **PS**.


Quando eseguo il comando **PS** comparirà sempre il processo del comando **PS** anche se quest'ultimo è già tecnicamente finito.

Possiamo usare il comando **psfux** per vedere quale processo è genitore di un altro processo.

 Ogni comando che eseguiamo crea un processo.

Esistono due tipi di processi:

- **Foreground:** Processi che vengono eseguiti sulla riga di comando che sto usando.
- **Background:** Ci serve per staccare dalla bash i processi in foreground, magari molto lunghi, in modo tale che possiamo continuare ad utilizzare la bash.

 Il comando **KILL** **NON** uccide il processo, ma manda un segnale. esistono diversi segnali che saranno descritti negli appunti successivi.

 **SSH:**

**Secure Shell**, è un protocollo di rete utilizzato per l'accesso sicuro a un computer remoto.