Is - [option ] [file/dirediory\_name ] Informazioni su file /directory .

* **-I** lista le info di una directory .
* **-f** Ulteriori informazioni associate ai file .
* **-a** Da informazioni anche sui file nascosti ( che iniziano con " . " “..” )
* **-A** come -a ma escludendo i file " . " e " . . " .
* **-F** Visualizza i file eseguibili con il suffisso " \* " e le directory con il suffisso "/" .
* **-d** directory.name Mostra le informazioni di directoryname ( come fosse un file) , senza mostrarne il contenuto .
* -**r(recent):** lista i file/directory in modo opposto all' ordine alfabetico
* **-R(recursive)** lista Ricorsiva .
* **-i** Mostra gli i - number.
* **-v** sort naturale
* **-t** lista i file/directory per ultima modifica (dal più recente al meno recente )

**Comandi semplici**

* **pwd**: mostra il percorso della dir corrente
* **cat:** mostra il contenuto di un file
* **mv source dest:** muove il file/dir nella dest specificata
* **cp sorg dest**:copia un file/dir nella dest e nome specificata
* **mkdir dir**:crea una dir
* **touch filename:**aggiorna la data di modifica del file
* **rmdir dir**: rimuove una dir se è vuota
* **rm -r** : per cancellare una directory anche se è piena
* **rm -i(interactive):** chiede esplicitamente se si vuole eliminare il file/dir
* **id**: indica Id,Gid,Group dell’utente
* **which**: sapere dove si trova un comando
* **whereis**: più info rispetto al comando which
* **who**: visualizza quanti utenti sono connessi
* **w:**come who ma fornisce info diverse

**Link Software & Hardware**

* **ln -s sorgente destinazione/nomelink:**  link software, è necessario che la sorgente sia un percorso assoluto. è possibile fare il link delle directory oltre a quello dei file
* **ln sorgente destinazione/nomelink:** link hardware, non è necessario che la sorgente sia un percorso assoluto, ma non è possibile fare un collegamento hw con una directory

**Sostituzioni**

* **\*:**  si sostituisce a tutti o a nessun carattere
* ?: si sostituisce ad almeno un carattere

per rendere la shell verbosa sh -x

**Aggiungere protezione ai file**

versione simbolica: **chmod u/g/o +/- r/w/x nomefile/nomedir**

versione ottale: **es: chmod 640 nomefile**

****

**( 6=110** al proprietario viene dato diritto di lettura e scrittura **4=100** al gruppo viene dato il diritto di lettura **0=000** agli altri nessun diritto**).**

**nb: se si vuole aggiungere qualche diritto è necessario ripetere i diritti già esistenti**

**Comandi Filtro**

* **pwd > t:** viene creato un file di nome t che contiene appunto il percorso del file con un a capo
* **ls -l >>(append) t:**  aggiunge caratteri al file t.

**Cat**

* **cat (senza niente):** il comando sta attendendo che l’utente scriva qualcosa sullo std. input. dopo aver scritto mi fornisce quello che ho scritto su std. output
* **ctrl-d**: rappresenta l**’eof** da input da tastiera.
* **cat > nomefile:**quello che viene digitato da tastiera finirà nel file
* **cat <nomefile**: visualizzo su std.output visualizzo il contenuto del file
* **cat <nomefile >nomefile1:** copia il contenuto del primo file nel secondo
* **more nomefile**: mostra contenuto del file
* **file eseguibile >nomefile/ file eseguibile** <nomefile: funziona come il cat

**Sort**

* **sort < nomefile:**  visualizza il file in ordine alfabetico
* **sort -r < nomefile** fa visualizzare l’ordine alfabetico al contrario
* **sort -f < nomefile**: ignora la differenza delle lettere maiuscole e minuscole
* **sort -c < nomefile:** viene riportata la prima riga in disordine.
* **sort -C < nomefile:**ci controlla se il file è ordinato.apparentemente sembra che non sia successo nulla (comando muto) .digitando **echo $?** ci restituisce o 1 se il file non era ordinato, 0 se il file era ordinato.
* **sort -u <nomefile:** fa visualizzare il file ordinato senza elementi doppi.

**Grep**

consente di cercare una certa stringa nello standard input. riporta tutte le righe dove c’è almeno una occorrenza della stringa richiesta

* **grep -n linea da cercare < nomefile**: visualizza anche la posizione delle linee trovate
* **grep -i linea da cercare < nomefile:** ignora la differenza tra lettere maiuscole e minuscole
* **grep -v linea da cercare < nomefile:** riporta tutte le linee che non contengano la parola cercata (ritorna l’opposto).
* se volessimo cercare una parola con degli spazi dobbiamo mettere la parola tra **‘apici ‘ (e lo spazio).** è importante la scelta di chiudere o lasciare aperte l’apice finale.
* **grep ‘^linea da cercare’< nome file:** trova tutte le linee che **cominciano** con la linea desiderata.
* es: **grep ‘^che’ < nomefile**: voglio trovare tutte le linee che incominciamo con che, in questo modo escludo le linee che hanno un che in mezzo alla linea o compreso in un altra parola.
* **grep ‘linea da cercare$’< nome file:** trova tutte le linee che **finiscono** con la linea desiderata.

se volessimo trovare il punto dobbiamo aggiungere il carattere di escape **\ (vengono utilizzati per tutti i metacaratteri)**.

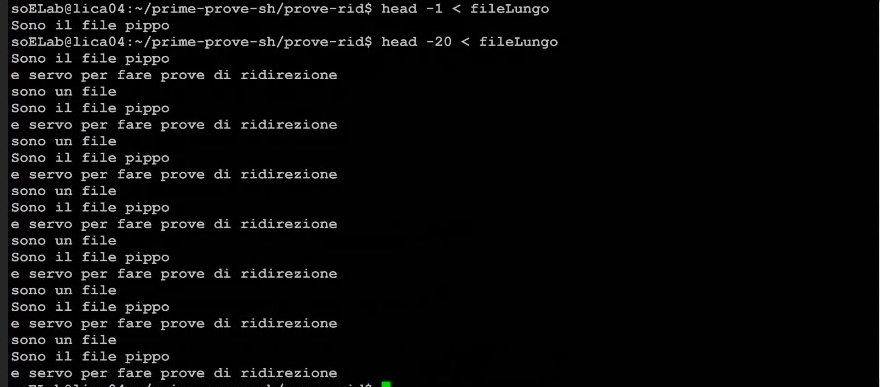
-**rev:** rovescia le linee dello standard input

**WC**

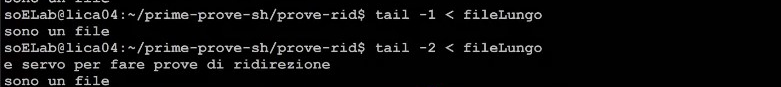
* **wc (word count):** conta tutte le parole
* **wc < nome file**
* 63 linee, 294 parole, 1470 caratteri. se volessimo solo una di queste info, si usano
* **wc -l < nome file (solo linee)**
* **wc -w < nomefile (solo word)**
* **wc -c < nomefile (solo caratteri)**

**Head & Tail**

* **head:** filtra le prime linee dello standard input
* senza specificare nulla il comando head riporta le prime 10 linee, se il file è più corto lo visualizza tutto



* **tail:**filtra le ultime linee dello standard input:



* senza niente fa visualizzare le ultime 20 linee
* **tail -1 < nomefile** :fa visualizzare l’ultima linea
* **tail -2 < nomefile**:le ultime due.

**redirezione dello Standard Error**

**2> /dev/null:**  gli eventuali errori dello standard error sono rediretti a null

es: ls -l z\* p\* >totale **2> &1:** l’output viene rediretto nel file totale. lo std. error(2) viene rediretto (>) come (&) lo std output (1)

**Pipe**

es ls -l | grep ‘^d’

**Altri comandi**

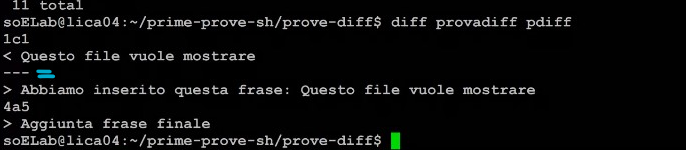
**-tee**: riporta tutto quello che è presente nello std in sia nel file desiderato e di visualizzare il contenuto anche da std output

per eseguire un processo in background si usa il & al termine del comando

es:**ls -lR / >temp 2> temperror &**

-**date**: restituisce la data del sistema

**-diff:** confronta due file e riporta linea per linea le differenza tra i file. il risultato del comando diff dipende dall’ordine di come mettiamo i file.



**<**: indica che è del primo file, **>**del secondo file

se invertiamo l’ordine:

opzione sort:

**-m(merge**)

**-o**(portate il risultato del sort in un file)

per debuggare un file comandi: **sh -x nomefile.sh**

riporta le linee eseguite dallo script

**env:** serve per visualizzare le variabili d’ambiente shell

**-unset:** elimina una variabile, se era una var di ambiente sparisce per sempre, se era una var semplice viene eliminata.