1. COMANDI LINUX BASE

- **ls** (list) → mostra il contenuto di una directory
- **ls -l /** → il carattere / va a indicare la root del file system
- **cat [nome file]** → serve per visualizzare il contenuto di un file di testo
- **cat /etc/passwd** → visualizza il contenuto del file /etc/passwd
- **cat shadow** → visualizza il contenuto del file shadow
- **cd** → consente di cambiare la directory corrente e scritto senza parametri ritorna alla HOME directory
- **pwd** → mostra il percorso fino alla directory corrente
- **id** → comando per visualizzare le informazioni UID e GID dell'utente
- **echo $HOME** → con questo si può visualizzare la HOME directory
- **ps** → visualizza i processi attivi dall'utente (PID → Process IDentifier, TTY → terminale, TIME → hh:mm:ss tempo di CPU e CMD → comando eseguito)
- **ps -f** → l'opzione -f (FULL) serve per avere maggiori informazioni sui processi
- **ps -fl** → visualizza informazioni ancora più estese -l(LONG)
- **ps -fle** → visualizza tutti i processi attivi nel sistema
- **man [comando]** → serve per sapere cosa fa esattamente un comando e quali opzioni prevede (man: MANUAL) (per procedere alle pagine successive si deve usare la barra spaziatrice; per procedere una riga alla volta si deve usare il tasto invio)
- **which [comando]** → serve per sapere dove si trova un comando
- **whereis [comando]** → serve per sapere anche dove si trova il manuale del comando specificato
- **sh** o **bash** → comando per invocare un'altra shell (dopo l'invocazione si può eseguire un altro comando e per terminare il processo della sotto-shell si deve utilizzare il comando exit; inoltre, si può invocare ulteriori shell tramite il comando "**bash**")
- **who** → comando per visualizzare gli utenti attivi in un sistema UNIX/LINUX
- **w** → visualizza più informazioni rispetto al comando "**who**"
- **ls -A** → visualizza i file con nomi che iniziano con il carattere "."
- **ls -a** → visualizza i file con nomi che iniziano con il carattere ".."
- **cd..** → serve per tornare alla directory precedente
- **cd ../..** → serve per tornare indietro di 2 directory
- **ls -R** → serve per vedere le sotto-gerarchie della directory corrente -R (RICORSIONE)
- **ls -F** → nella visualizzazione di file e directory, viene aggiunto il carattere / dopo il nome se è una directory
- **ls -FR** → nella visualizzazione viene aggiunto il carattere * se è un file eseguibile
- **ls -FlR** → visualizza la sotto-gerarchia della directory corrente facendo vedere tutti i dettagli, anche sul tipo del file
- **ls -r** → visualizza il contenuto di una directory in ordine alfabetico inverso -r (REVERSE)
- **ls -t** → vengono mostrati i file in ordine di ultima modifica
- **ls -tl /home/cesare** → visualizza il contenuto della propria HOME directory partendo dal file modificato più recentemente
- **touch [nome file]** → serve per modificare la data di un file

- **ls -ld [nome directory]** → serve per vedere le informazioni relative al file che rappresenta la directory specificata e il numero di link
- **ls -l [nome file]** → serve per avere informazioni solo sul/sui file specificati
- **ls -l *ile** → l'asterisco indica una stringa prima dei caratteri "ile" e visualizza quindi tutti i file che terminano con "ile"
- **ls -l ?ile** → il punto interrogativo indica un carattere prima dei caratteri "ile" e visualizza quindi tutti i file che terminano con "ile"
- **ls – l file?** → serve per avere informazioni sui file che hanno come nome il nome "file" seguito da un carattere
- **ls -l file*** → serve per avere informazioni sui file che hanno come nome il nome "file" seguito da una stringa
- **ls *t*** → visualizza i file che hanno nel nome una "t"
- **sh -x** → invoca un sotto-shell consentendo anche di vedere che cosa espande la shell prima di invocare il comando dove sono stati usati dei metacaratteri **(* e ?)**; dopo l'invio del comando appare il simbolo "+" all'inizio della riga dove viene mostrato cosa espande/sostituisce la shell prima di eseguire il comando
- **echo [stringa]** → stampa la stringa su standard output
- **echo file*** → uso del pattern matching sul contenuto della directory corrente, quindi stampa su standard output i file il cui nome inizia per "file"
- **ls *f?** → verifica le sostituzioni del pattern matching
- **echo *p?** → stessa cosa ma viene visualizzato sulla standard output
- **ls -l /etc/passwd /etc/shadow** → visualizza i diritti di accesso dei file /etc/passwd e /etc/shadow
- **vi [nome file]** → è un editor di testo e serve per creare file
- **chmod u-w [nome file]** → elimina per l'utente corrente il diritto di scrittura al proprio file specificato (con il "+" invece viene aggiunto)
- **chmod u-r [nome file]** → elimina per l'utente corrente il diritto di lettura al proprio file specificato (con il "+" invece viene aggiunto)
- **chmod u-x [nome file]** → elimina per l'utente corrente il diritto di esecuzione del proprio file specificato (con il "+" invece viene aggiunto)
- **chmod 600 [nome file]** → ripristina i diritti di lettura e di scrittura per l'utente togliendo qualunque altro diritto a gruppo e altri
- **mkdir [nome directory]** → serve per creare una nuova directory
- **rmdir [nome directory]** → serve per cancellare una directory
- **rm [nome/nomi file]** → serve per cancellare uno/più file
- **rm -i [nome file]** → serve per cancellare uno/più file chiedendo la conferma da parte del S.O
- **rm -ri [nome directory]** → cancella ricorsivamente file e directory chiedendo la conferma da parte del S.O.
- **ln [nome file] [nome directory]/[nome link]** → serve a creare un link hardware ad un file nella cartella specificata; quindi il file sarà ora accessibile tramite due nomi assoluti
- **ls -li [nome file]/[nome del link]** → serve per visualizzare sia l'i-number che l'inode che il numero di link

- **cp [nome file] [nome directory]/[nome copia]** → crea la copia di un file nella directory specificata; con questo comando il file originario e la copia hanno gli i-number diversi e quindi gli i-node diversi
- **mv [nome file] [nome file rinominato]** → rinomina un file
- **mv [nome file] [nome directory]/[nome file]** → sposta il file specificato nella directory specificata
- **ln -s [nome file] [nome directory]/[nome link]** → crea un link software di un file nella directory specificata
- **cat *** → stampa a video il contenuto di tutti i file presenti nella directory corrente

2. COMANDI RIDIREZIONE

- **cat > [nome file]** → crea un file su standard input e lo si compila; per uscire dallo standard input bisogna fare CTRL-D
- **pwd > [nome file]** → inserisce nel file il contenuto del comando "pwd", ma non lo aggiunge
- **pwd >> [nome file]** → aggiunge al contenuto del file ciò che viene visualizzato eseguendo il comando "pwd" (ridirezione in append)
- **ls -l >> [nome file]** → aggiunge al contenuto del file ciò che viene visualizzato eseguendo il comando "ls -l" (ridirezione in append)
- **cat < [nome file] > [nome della copia del file]** → crea una copia del file e gli assegna un nome scelto dall'utente
- **cat [nome file] >> [nome altro file]** → aggiunge il contenuto del file specificato per primo al file specificato per secondo (ridirezione in append)
- **more < [nome file]** → visualizza il contenuto del file come cat (è più utile per i file molto lunghi
- **more < f*** → visualizza il contenuto di tutti i file che iniziano con la lettera "f"; viene mostrato un file alla volta e per procedere bisogna usare la barra spaziatrice
- **sort < [nome file]** → serve per ordinare il contenuto del file specificato
- **sort < [nome file] > [nome file ordinato]** → ridirige l'ordinamento su un altro file
- **sort -r < [nome file]** → serve per ordinare il contenuto del file specificato in modo inverso
- **sort -f < [nome file]** → serve per l'ordinamento ignorando maiuscole e minuscole
- **sort -fr < [nome file]** → combinazione
- **sort -c < [nome file]** → riporta la prima linea che non verifica l'ordinamento alfabetico (-c: CHECK)
- **echo $?** → riporta il valore di ritorno che, in caso fosse 1, indica un insuccesso, 0 altrimenti.
- **sort -C < [nome file]** → stesso risultato di -c, ma non scrive nulla sullo standard output; quindi, è necessario eseguire il comando "echo $?"
- **sort -c < [nome file ordinato] 2> /dev/null** → verifica l'ordinamento del file specificato, ridirigendo lo standard error su /dev/null

- **sort -u < [nome file]** → ordina il contenuto del file eliminando le linee ripetute
- **grep [stringa/carattere da cercare] < [nome file in cui cercare]** → serve per cercare dei pattern nei file
- **grep -n [stringa/carattere da cercare] < [nome file in cui cercare]** → riporta il numero della linea dove è stata trovata la stringa che si sta cercando
- **grep -i [stringa da cercare] < [nome file in cui cercare]** → serve per ignorare la differenza tra maiuscole e minuscole
- **grep -v [stringa da cercare] < [nome file in cui cercare]** → serve per invertire la ricerca, cioè vengono mostrate solo le linee che NON contengono il pattern
- **grep '^[stringa]' < [nome file in cui cercare]** → ricerca tutte le linee che iniziano per una certa stringa
- **grep '^[carattere]' < [nome file in cui cercare]** → ricerca tutte le linee che iniziano per un certo carattere
- **grep '[carattere]$' < [nome file in cui cercare]** → ricerca tutte le linee che finiscono per un certo carattere
- **grep '[stringa]$' < [nome file in cui cercare]** → ricerca tutte le linee che finiscono per una certa stringa
- **grep '\.$' <** [nome file in cui cercare] → ricerca tutte le linee che finiscono per il carattere "." (punto)
- **wc < [nome file]** → conta le linee, le parole e i caratteri di un file di testo (wc: WORD COUNT)
- **wc -l < [nome file]** → riporta solo le linee di un file di testo
- **wc -w < [nome file]** → riporta solo le parole di un file di testo
- **wc -c < [nome file]** → riporta solo i caratteri di un file di testo
- **wc [nome file]** → riporta anche il nome del file, oltre al numero di linee, parole e caratteri
- **head < [nome file]** → mostra di default le prime 10 linee di un file di testo
- **head -1/-2/-3 < [nome file]** → si specifica il numero di linee che devono essere mostrate
- **tail < [nome file]** → mostra le ultime 10 linee di un file di testo
- **tail -1/-2/-3 < [nome file]** → si specifica il numero di linee che devono essere mostrate
- **head -3 < [nome file] > [nome file head]** → si ridirige lo standard output, in questo caso le prime 3 linee, sul file specificato come secondo parametro
- **tail -3 < [nome file] > [nome file tail]** → si ridirige lo standard output, in questo caso le ultime 3 linee, sul file specificato come secondo parametro
- **rev < [nome file]** → rovescia i caratteri presenti, linea per linea, in un file di testo

2.1 RIDIREZIONE DELLO STANDARD OUTPUT/ERROR

- **> [nome file]** → creazione di un file con la ridirezione a vuoto

- **ls -l** [carattere]* [carattere]* **>** [nome file] → in questo modo viene ridiretto solo lo standard output e, se non sono presenti file che hanno come iniziale uno dei caratteri specificati, verrà mostrata la linea: ls: cannot access [carattere]*: No such file or directory → questa linea viene scritta sullo standard error, mentre nel file specificato saranno presenti tutti gli altri file che hanno come iniziale l'altro carattere
- **ls -l** [carattere]* [carattere]* **2>** [nome file] → in questo caso viene ridiretto solo lo standard error e il file specificato conterrà la linea: ls: cannot access [carattere]*: No such file or directory
- **ls -l** [carattere]* [carattere]* **2> /dev/null** → se lo standard error non interessa, si può ridirigere sul dispositivo /dev/null
- **ls -l** [carattere]* **[carattere]* > [nome file] 2>&1** → in questo caso viene ridiretto sia lo standard output che lo standard error sullo stesso file

3. PIPING

- **ls -l | grep '^d'** → trova tutte le linee che corrispondono a nomi di directory
- **ls -l | grep '^d' | wc -l** → trova tutte le linee che corrispondono a nomi di directory e le conta
- **ls -l | grep '^d' | tee** [nome file] **| wc -l** → per tenere traccia di cosa si è trovato, si utilizza il comando "tee" che permette di creare un file, al cui interno viene riportato il risultato del comando utilizzato
- **ps | tee** [nome file] **| wc -l** → mostra il numero dei processi per ogni comando in piping e il contenuto viene inserito all'interno del file creato dal comando "tee"
- **head -3 <** [nome file] **| tail -1 >** [nome file] → utilizzando il piping, si isola nel file specificato come secondo parametro la terza linea a partire dall'inizio del file specificato come primo parametro
- **tail -3 <** [nome file] **| head -1 >** [nome file] → utilizzando il piping, si isola nel file specificato come secondo parametro la terza linea a partire dalla fine del file specificato come primo parametro

4. ESECUZIONE IN BACKGROUND

- **ls -lR / >** [nome file] **2>** [nome file] **&** → tramite questa scrittura viene mandato in esecuzione, in background, il comando ls -lR e il contenuto viene inserito nel file specificato come primo parametro. Per terminare l'esecuzione in background bisogna utilizzare il comando "kill"

5. ALTRI COMANDI

- **date** → mostra la data e l'ora attuale
- **diff** [nome file] [nome file] → consente di verificare le differenze fra due file. Mostra cosa ha in più o in meno il file passato come secondo parametro rispetto al file passato come primo parametro
- **find . -name** [stringa] → si va a ricercare all'interno della directory corrente tutti i file con nome relativo semplice quello specificato nel campo [stringa]

- **find . -name *[stringa]*** → si va a ricercare all'interno della directory corrente tutti i file il cui nome relativo semplice contenga la stringa specificata nel campo [stringa]

6. VARIABILI

- **a=10** → definiamo una variabile di shell di nome "a" e gli assegniamo il valore 10
- **echo $a** → visualizza il valore della variabile di shell di nome "a"
- **a=ciao** → cambiamo il valore della variabile di shell di nome "a"
- **x=`wc -l < [nome file]`** → memorizzare in una variabile di nome x il numero di linee del file specificato. Si usano gli apici rovesciati (**ALT GR + APOSTROFO)**
- **z=`expr $x + $y`** → memorizza in z la somma della variabile x e della variabile y. Si usano gli apici rovesciati (ALT GR + APOSTROFO)
- **echo $HOME** → variabile di ambiente HOME
- **echo $PATH** → variabile di ambiente PATH che, per cercare il file eseguibile da mandare in esecuzione alla richiesta di un comando, deve usare il cammino ".", che rappresenta la directory corrente. Quindi per mandare in esecuzione un file eseguibile bisogna scrivere **"./[nome file eseguibile]"**
- **env** → comando per visualizzare tutto l'ambiente (variabili di ambiente)
- **export [nome variabile]** → rende la variabile di shell specificata una variabile d'ambiente. La modifica di una variabile di ambiente effettuata da una sotto-shell ha effetto solo in quel processo e non nel processo padre
- **unset [nome variabile]** → viene eliminata la variabile di shell o di ambiente specificata

7. METACARATTERI

- **echo [abc]*** → stampa i file che iniziano per a,b o c
- **echo [a-z]*** → stampa i file che iniziano per un carattere compreso tra a e z
- **echo [A-Z]*** → stampa i file che iniziano per un carattere MAIUSCOLO
- **echo [0-9]*** → stampa i file che iniziano per un carattere numerico
- **echo *[abc]** → stampa i file che terminano per a,b o c
- **echo *[a-z]** → stampa i file che terminano per un carattere compreso tra a e z
- **echo *[A-Z]** → stampa i file che terminano per un carattere MAIUSCOLO
- **echo *[0-9]** → stampa i file che terminano per un carattere numerico

Con il **punto esclamativo "!" si fanno le negazioni** (es. echo [!abc]* stampa tutti i file che NON iniziano per a,b o c)

Con il **comando "ls -l" seguito dai metacaratteri sopra elencati si visualizzano anche le informazioni inerenti ai file e, nel caso di directory, anche il loro contenuto.**

- **echo *** → il carattere **escape "\"** serve per negare il significato di metacarattere

8. SOSTITUZIONI

- **Sostituzioni di variabili:**

```
cesare@cesare-VirtualBox:~/Esercizi$ ls
prova  prova-bis.sh  prova.sh  p.txt  t
cesare@cesare-VirtualBox:~/Esercizi$ a=pippo
cesare@cesare-VirtualBox:~/Esercizi$ echo $a
pippo
cesare@cesare-VirtualBox:~/Esercizi$ ls -l $a
ls: impossibile accedere a 'pippo': File o directory non esistente
cesare@cesare-VirtualBox:~/Esercizi$ a=t
cesare@cesare-VirtualBox:~/Esercizi$ echo $a
t
cesare@cesare-VirtualBox:~/Esercizi$ ls -l $a
-rw-rw-r-- 1 cesare cesare 22 apr 17 11:36 t
```

- **Sostituzioni di comandi:**

```
cesare@cesare-VirtualBox:~/Esercizi$ echo `pwd`/$a
/home/cesare/Esercizi/prova
cesare@cesare-VirtualBox:~/Esercizi$ ls -l `pwd`/$a
-rw-rw-r-- 1 cesare cesare 31 apr 17 11:37 /home/cesare/Esercizi/prova
```

- **Sostituzioni di nomi di file:**

```
cesare@cesare-VirtualBox:~/Esercizi$ ls -l `pwd`/$a*
-rw-rw-r-- 1 cesare cesare  31 apr 17 11:37 /home/cesare/Esercizi/prova
-rwxrw-r-- 1 cesare cesare 232 apr 17 12:48 /home/cesare/Esercizi/prova-
bis.sh
-rwxrw-r-- 1 cesare cesare 169 apr 17 12:34
/home/cesare/Esercizi/prova.sh
```

- **eval $[nome variabile]** → serve per forzare la shell ad attuare di nuovo le sostituzioni