echo ciao & date : la & esegue entrambi i comandi invece di eseguire prima uno e poi un'altro, mentre la doppia && commerciale è un AND

; : carattere speciale per la separazione dei comandi su una stessa riga senza andare a capo.

test -f nomefile : verifica esistenza del file.

echo $? : il risultato dell'ultimo comando effettuato.

&> devnull : reindirizza l'output indesiderato

printenv : stampa l'enviroment.

bin/bash : chiama una sottoshell.

SHLVL : si ottiene con printenv, indica il livello di shell in cui siamo. SHLVL = 1 è la shell di partenza.

whereis <comando> : stampa percorso del comando.

cmp : compara due file byte by byte.

sort : ordina in ordine alfabetico o crescente

uniq : elimina doppi di valori o stringhe adiacenti

sort <file> | uniq : ordina e poi rimuove i doppi

sort <file> | uniq -c | sort -n : fa il sort del contenuto del file, lo passa a uniq che estrare I valori una volta sola con conteggio delle ripetizioni, e viene passato a sort numerico

find ./ -name '*E*' -or -name 'F*' : trova tutti I file che cominciano con F oppure hanno la E

find ./ -iname '*E*' -or -iname 'F*' : trova tutti i file che cominciano con F maiuscola o minuscola oppure hanno la E maiuscola o minuscola

find ./ -type f -size +100k : trova files più grandi di 100k

find ./ -type f -size +100k -size -1M : trova files di grandezza superiore di 100k ma minore di 1M

find ./ -type f -exec cat {} \; : trova i files e per ognuno concatenali. Il comando va terminato con \;. Mentre le parentesi graffe sono il place holder per I risultati.

find ./ -type f -size + 50M 2>/dev/null -exec ls {} \; : trova tutti I file più grandi di 50M, gli errori mandali a dev/null, esegui listing sul place holder speficicato dalle parentesi graffe.

grep : è un tool importante che cerca ricorsivamente valori/testo nei files. E' case sensitive

grep 'dropbear' <file> : ricerca e trova la parola dropbear nel file specificato

awk : crea filtri per eseguire script che accettano dati stdin per poi cambiarli ed indirizzarli in stdout.

awk ‘{print $1,$3}’ file.txt : prende in input i dati di file.txt e manda in output la prima colonna e la terza.

awk -F’:’ ‘{print $1}’ /etc/passwd : dal file passwd tramite il separatore -F separiamo l’output con i doppi punti : e stampiamo solo la prima colonna.

sed -i ‘s/pizza/pasta/’ file.txt : stream editor. Cambia qualsiasi string (s) nel file da ‘pizza’ a ‘pasta’. (-i) rende il cambiamento definitivo.

sed -i ‘s./etc..’ file.txt : rimuove tutte le occorrenze che hanno ‘/etc’. Il punto (.) dopo (s) diventa il nuovo delineatore. Possiamo assegnare qualsiasi delineatore.

du : fa il display di tutti I file con relativo disk usage

du -kh /home : disk usage dei file di home in kilobytes human readable

du -mk /dev : disk usage dei file di dev in megabytes human readable

du -h | sort -h | tail -n5 : disk usage in human readable che viene passato a sort crescente in human readable, di cui prelevo i 5 più grandi.

df -h : informazioni sui file system montati.

df -h home/user : informazioni su /home compreso file system, grandezza ed utilizzo

history : stampa l’intera storia di comandi utilizzati. Utile da usare con grep.

ps : serve a ispezzionare o vedere lo stato del processo o dei processi avviati.

ps ax : visualizza TUTTi I processi avviati dalla macchina.

ps axww : il formato è completo della visualizzazione. Non è tagliato come ps ax

gzip : comprimere files

gzip filename : comprime il file stesso senza creare un dupplicato

gzip -k filename : crea una copia del file comprimendola. Si ha quindi originale (keep) e copia compressa.

gzip -kv filename : v sta per verboso. Mostra le info della compressione.

gzip -d filename.gz : decomprime il file compresso.

gunzip filename.gz : altra versione di decompressione.

tar : (tape archive) crea un archivio di più files in uno singolo

tar -cvf archivio.tar file1 file2 file3 : create verbose file il cui nome è srchivio.tar composto dai tre file successivi. Se si utilizza f in cvf bisogna specificare il nome dell'archivio. NON è compresso.

tar -tf archive.tar : mostra solo il contenuto dell'archivio.

tar -xvf archivio.tar : estrae I file dall'archivio nella cartella corrente.

tar -xvf archive.tar -C directory : estrae I file in una specifica directory.

find /usr/include/ -name 's*.h' -exec tar -rvf file.tar {} \; : trova i file dentro il path con I parametri specificati ed uno alla volta li aggiunge all'archivio file.tar

gzip -k archive.tar : comprime l'archivio creando una copia.

gzip -d archive.tar : decomprime l'archivio.

tar -cvzf archive.tar.gz file1 file2 : crea archivio con nome specificato con i file, e comprime il tutto. Per decomprimere ed aprire i file basta : **tar -xvzf archivio.tar.gz**

xargs : prende gli output di un comando e li trasforma in argomento input di un secondo comando.

cat list.txt | xargs rm : prende I valori di output di cat e uno ad uno gli rimuove con rm.

find ./ -size +10M | xargs ls -lah : trova i file più grandi di 1 mega dentro ./ in modo ricorsivo e li passa tramite xargs come argomento a lista.

su <utente> : posso cambiate utente bash

passwd : è possibie cambiaare passwords.

sudo passwd elvis : è possibile cambiare la password dell'utente elvis.

chown : è possibile cambiare chi possiede un file o directory.

sudo chown <new owner> file : sintassi per il cambio owner.

sudo chown -R <new owner> file : ritorna il precedente owner dei files e directories.

groups : lista tutti i gruppi a cui l'attuale user appartiene.

sudo chown <new owner>:<new group> file : cambio sia l'owner che il group di un file.

var1="ciao" : crea una varibile locale. Le variabili locali esistono SOLO nel livello shell creato.

unset var1 : distrugge la variabile.

export var1: la variabile da locale diventa globale. Con il comando **printenv** si possono localizzare le variabili globali.

set : controllo delle variabili nel sistema e locali. Indica anche il livello shell.

- : regular file
d : directorie
c : character special file
l : symbolic link

u : user owner
g : group owner
o : others
a : all the above

chmod g+w file.txt: si aggiunge al gruppo la possibilità di scrivere sul file.

chmod a-w file.txt : si toglie a tutti la possibilità di scrivere sul file.

chmod a+r file.txt : tutti possono leggere il file.txt.

chmod a-w file.txt : a tutti è vietato scrivere.

chmod o-rwx file.txt : tutti I permessi sono rimossi a others.

chmod a=r file.txt : tutti possono SOLO leggere. L'uguale risetta/rimuove gli altri permessi.

chmod u+s file.txt : modalità suid. Si attibuisce al file la modalità suid e anche uscendo da root il file

Ottenere numero di tutti i file *.h contenenti un numero di righe pari in una data directory
find /usr/include -type f -name '*.h' | xargs wc -l | sed 's/\s\+/ /g' | cut -d' ' -f2 | xargs -I{} expr {} % 2 | grep ^0 | wc -l

Cercare parola "Bash" in un file:
if grep -q Bash filename; then echo "Il file contiene almeno una parola di Bash."; fi

Cercare stringa in una parola (-q sopprime l'output che può essere molto grande):
parola=linux
seq=inu
if echo "$parola" | grep -q "$seq"; then echo "$seq trovata in $parola"; else echo "$seq non trovala in $parola"; fi

Crea un file di valori random in fileout con block size (bs) da 1024 bytes per velocizzare. Count delinea la grandezza del file, in questo caso 10M
sudo dd if=/dev/urandom of=fileout bs=1K count=10240

Bit by bit backup image:
dd if=/dev/sdb1 of=/dev/sda2 bs=1M conv=noerror

Misure:

Byte = 8 bits
KB = 2^10 = 1.024 bytes
MB = 2^20 = 1.048.576 bytes
GB = 2^32 = 1.073.741.824 bytes
TB = 2^40 bytes

secondo (s) = 1.000 (ms) – 1.000.000 (µs) – 1.000.000.000 (ns)
millisecondo (ms) = 0.001 (s) – 1.000 (µs) – 1.000.000 (ns)
microsecondo (µs) = 0.000001 (s) – 0.001 (ms) – 1.000 (ns)
nanosecondo (ns) = 0.000000001 (s) – 0.000001 (ms) – 0.001 (µs)