COMANDI

ls + nomefile/directory: (list) mostra le informazioni relative a file/directory.
Senza parametro: informazioni della directory corrente.

-l (long)	lista oltre al nome tutte le info del file
-a (all)	lista anche i nomi dei file nascosti (quelli che iniziano per ".")
-A	come -a ma esclude **.** e **..** (directory padre e directory corrente)
-F	lista i nomi dei file eseguibili con alla fine ***** e i nomi delle directory con alla fine **/**
-d	**+ nomefile/directory** mostra le informazioni della directory/file, non il contenuto
-R (cursive)	Parte dalla dir corrente (.) e elenca i nomi dei file contenuti, poi scende nell'albero e compie la stessa cosa per le directory sottostanti (contenute)
-i	mostra anche l'i-number (numero ls
-r	lista i nomi dei file in ordine alfabetico opposto (compie l'opposto di -ls)
-t	li mette in ordine di ultima modifica

possono essere anche composti per es: *ls -la* (mostra i nomi le info + i file nascosti)

cat + nomefile: (catenate) visualizzare contenuto di un file
Con Più PARAMETRI concatena in output il contenuto dei file
Comando principale per leggere i file di testo

- *cat /etc/passwd* uso il NOME ASSOLUTO(ogni nome assoluto inizia con /) del file
- *cd /etc* Mi sposto nella directory giusta e uso il
  cat passwd

more+nomefile: come il cat
more+nomefile1+nomefile2: mostra il contenuto dei due file concatenati in output come il cat, ma aggiunge una intestazione con il nome del file prima di mostrare il contenuto

sudo cat shadow: per visualizzare il contenuto di un file protetto
Se scrivo solo cat shadow ho permisson denied perché shadow appartiene non a noi ma al superutente

cd + nomedirectory: (change directory) per spostarmi nelle directory (cambia la mia directory corrente),
cd senza parametri torna alla home
Cd $HOME Variabile di ambiente che ci riporta comunque a casa

pwd: (printing working directory) mi dice in che directory sono, stampa il nome della dir corrente(working directory)

id: visualizzare le info di UID e GID dell'utente corrente

who: mostra gli utenti collegati al server
w fa la stessa cosa ma da ulteriori informazioni

echo+stringa: visualizza la presenza della stringa all'interno del nome dei file
(* e ? per indicare un carattere qualsiasi, * può anche non esserci, ? deve per forza essere un carattere)
echo $HOME dice qual è la home dir, perché ritorna il valore della variabile
Echo *ile ritorna i nomi di file che terminano in ile come stringhe

ps: (process status) mostra i processi attivi nella sessione

-f (full)	Ci dà piu informazione come l'UID

-l	Mostra lo status del processo (running,
-e	Ci fa vedere l'elenco dei proc con le info di base

man +comando: ci mostra il manuale del comando
which+ comando : ci dice dov'è il comando nel file system
whereis+ comando: ci dice la stessa cosa di which ma anche dov'è stato reperito il manuale del comando

touch: va a "toccare" il file, cambia la data di ultima modifica del file a quella attuale
Serve anche per creare file vuoti
touch pippo crea un file vuoto di nome pippo

sh: per invocare una shell
Sh -x xtrace guarda slide 8
Che tipo di espansione e comando viene effettuato
Stampa i comandi degli argomenti
Exit: uscire dalla shell processi

vi + nomefile:	creare un file
wq	salvare e uscire dal file
q!	uscire senza salvare dal file
i	entrare nella fase insert
a	entrare nella fase modifica
esc	tornare al display comandi nel file
x	cancellare un carattere
dw	cancellare parola
dd	cancellare linea
Ctrl c	Uscire dalla fase modifica/insert

chmod+ opzioni+ nomefile: (change mode) cambia i diritti su un file/directory

- *chmod g+r provadiritti* (voglio aggiungere al gruppo la possibilità di read)
  In questo modo modifico solo un bit
  chmod g-r provadiritti per togliere il diritto di read al gruppo
  chmod +r provadiritti aggiungo il diritto di read a tutti
- *chmod 600 provadiritti* (6 in ottale =110, 0=000,0=000) metto diritto di r e w nell'utente proprietario e il resto rimane a 0
  In questo modo devo specificare e modificare tutti i bit

UTILIZZABILI SOLO DAL SUPERUTENTE
chown+ nomeutente+ nomefile: (change owner) cambia il proprietario di un file/directory
chgrp+ nomegruppo+ nomefile: (change group) cambia il gruppo di un file/directory

ln + nomefile_originale+ percorso_nuovo: (link hardware) crea un percorso differente per arrivare alle stesse informazioni
NON è UNA COPIA
ln provalink dir/p crea nella directory dir un descrittore p che rimanda al contenuto del file provalink
LINK SOFTWARE

ln -s +percorsodelfileintero+ percorsonuovo	per creare link software e non hardware

ln -s /home/marianna/provalink linksoftware/pl sto creando un link software di provalink nella cartella linksoftware e lo chiamo pl
Il link originale deve essere sempre assoluto
Viene creato un file ad hoc che contiene la stringa con il nuovo percorso assoluto

mkdir +nomedir: (make directory) per creare una nuova directory nella directory in cui mi trovo

cp+nomefile+percorso(directory+nuovonome): (copy) creare una copia di file
cp provalink dir/paperino creo una copia di provalink in dir e la chiamo paperino

rm +percorso/nomefile: (remove) decrementa il numero dei link, se poi il num di link è 0 allora vengono eliminate le informazioni dal sistema e restituito l'i-node. Se invece il numero di link non è 0, allora verrà solo cancellato il link
si consiglia sempre di usare

rm -i (interactive)	in modo che chieda conferma della cancellazione, cosi non cancello per sbaglio
rm -ir (recorsive)	per cancellare le directory anche da piene

rmdir + nomedirectory: cancella una directory SOLO se è vuota
Per rimuovere directory piene o gerarchie di directory devo usare rm -ir

mv+ nomefile+ percorsodovespostare: (move) consente di spostare un file, è un ln+rm crea un nuovo link e rimuove quello precedente
mv plink ../nuovo/p sapendo di trovarmi in dir sposto il mio file nella cartella nuovo e lo chiamo p
mv paperino pippo può essere usato anche come rename

SI Può USARE Più DI UN COMANDO CON IL ";"

RIDIREZIONE
comando+ > + nomedovevogliostampare: ridirezione standart output del comando
se lo uso senza comando creo un file vuoto
comando+ < + nomedovevoglioleggere: ridirezione input del comando
esempio

cat come filtro	Prende un parametro in input e lo scrive in output
ctrl + d	terminare il comando in modo corretto, terminare il cat
cat > t1	Input(nostra tastiera) output(t1): permette di scrivere dentro il file t1
cat < t1	Input(t1) Output(schermo): mi fa vedere cosa ho scritto in t1
cat < t1 > t2	scrivo in t2 cosa ho scritto in t1 (RENAME?)

Cat: senza parametri prende come input la tastiera e output lo schermo quindi tutto quello che digitiamo viene riscritto a capo (per uscire faccio ctrl d)
>>: attacca in append senza cancellarti nulla, se uso solo > sovrascrivo
>+ nomefile: RIDIREZIONE A VUOTO--> crea un file di lunghezza 0, vuoto

gcc+ nomefile : effettua compilazione e linking del file che passo e mi mette l'output in un file a.out
gcc -o prova prova.c: come prima ma l'output me lo mette in prova. Adesso prova è un filtro quindi scrivo in input e me lo da in output

more+ </>+ nomefile: il more su file corti non ha molta differenza dal cat, ma quando il file è più lungo delle righe visualizzabili in un terminale testuale. Mi fa vedere dolo in parte il contenuto poi scrive more. Per vedere ulteriori righe ENTER, per ulteriori pagine SHIFT, per uscire q
more < file lungo
cat t riporta il concatenamento del contenuto dei file*
ciò non può essere replicato se usiamo la ridirezione
non possiamo agganciare più file allo standard input
cat < t1
more <t1
lo stesso vale per il more, quindi se non è un comando filtro possiamo scrivere
more t: tutti i file il cui nome comincia per t verranno visualizzati, ma solo quelli testo, gli altri verranno segnalati come non di testo. more ci da quindi più sicurezza di protezione.*

sort+</>+ nomefile: mette in ordine alfabetico il contenuto (sia come filtro che no)
Segue il codice ASCII: prima le righe vuote, poi prima quelle con le maiuscole, e poi con le minuscole

sort -r	dispone in ordine alfabetico inverso
sort -f	ignora la distinzione tra maiuscole e minuscole
sort -c (check)	controlla se il file è effettivamente ordinato, se non lo è comunica "disorder" sullo schermo
sort -C	Come -c, ma non ritorna "disorder" (per vedere successo/insuccesso guardo $?)
sort -u (unique)	otteniamo il contenuto ordinato, ma se ci sono due righe uguali ne scrive solo una

sort < provasort> ordinato organizza il contenuto di provasort e lo mette nel file creato ordinato
SORT POTREBBE DARE PROBLEMI SULLA VIRTUALBOX NEL CASO CHIEDERE ALLA PROF

$?: variabile in cui vengono salvati i valori di ritorno dei comandi
I valori che può assumere sono

0	Significa SUCCESSO del comando
> 0	INSUCCESSO del comando

echo $?: visualizza il valore ritornato dall'ultimo comando.

grep+ pattern che sto cercando + </>+ nomefile: (general regular expression print) serve per cercare un pattern(una sequenza di lettere) all'interno di file di testo. Ritorna cosa è scritto nella riga in cui è presente il pattern.

grep -n	mi ritorna oltre al testo anche l'indice della riga in cui ho l'occorrenza
grep -i	ignora che la parola che sto cercando sia minuscola o maiuscola, ritorno senza fare distinzione
grep -v	ritorna il testo delle linee che non hanno il pattern
grep '^ pattern'	individua tutte le linee che cominciano con un certo pattern
grep 'pattern$'	individua tutte le linee che finiscono con un certo pattern
grep '\caratterediescape'	per cercare caratteri speciali come quelli di escape (come il punto) (lo slash prima del punto gli toglie significato e lo rende un carattere normale)

grep file <provagrep mi manda in stampa tutte le righe in cui c'è almeno un'occorrenza di 'file'
grep '\.$' provagrep ritorna le linee di provagrep che terminano con il punto

wc+</>+nomefile: ci da il numero di linee, il numero di parole e il numero di caratteri che sono contenuti in un file

wc -l (line)	ci mostra solo il numero di linee
wc -w (word)	ci mostra solo il numero di parole
wc -c (caracter)	ci mostra solo il numero di caratteri

il comando utilizzato non in ridirezione ha le stesse opzioni, mostra il numero e il nome di file, anche wc se non è un filtro ammette il nome di più files.
wc -c filelungo mi mostra il numero di caratteri e il nome di tutti i file che si chiamano fileLungo
wc -c <fileLungo mi mostra solo i numero di caratteri

head +</>+nomefile: stampa le prime dieci linee del file (o meno se ne esistono meno)
tail +</>+nomefile: stampa le ultime dieci linee del file (o meno se ne esistono meno)

head/tail -1	visualizza la prima/ultima linea del file

Se non va -1 sulla virtual box bisogna scrivere -n 1

rev+</>+nomefile: stampa il reverse del contenuto di ogni singola linea, le mostra scritte al contrario
Sono il file pippo--> oppip elif li onoS

1	Identifica lo standard input
0	Identifica lo standard output
2	Identifica lo standard error

comando+ 2> + nomefile: scrivo lo standard error del comando nel file (non voglio più visualizzarlo in output)
2 + /dev/null: quando voglio ignorare lo standard error lo ridireziono a null
Comando+ >&2: metto l'output del comando sullo standard error(2)
Comando > nomefile 2>&1*:* metto l'output nel file e nello standard error ho sia error che output
ls -l z p* >totale 2>&1* il risultato del comando ls finisce nel file 'totale', l'output finisce in standard error (2)
ls -l z p* 1>totale 2>&1* è la stessa cosa
--> a video non vedremo nulla, l'output lo troveremo nel file totale o in standard error

PIPELINE
| : simbolo per le pipe, l'output di un comando è l'input di quello dopo
ls -l | grep '^d' | wc -l della lista di file che leggo individuo le directory (iniziano con d) e le conto

tee+nomefile : si può usare solo come comando filtro, prende ciò che trova in input e lo manda in output scrivendolo contemporaneamente anche sul file specificato
Quindi crea o sovrascrive file.
ls -l | grep '^d' | tee tempdir |wc -l stessa cosa di prima ma in tempdir ho tutte le linee che iniziano per d (cioe in file che iniziano per d e tutte le loro info)
ps | wc -l da in output 4 perché, ps darebbe già 3 linee (etichette, dettagli processo bash, dettagli processo ps) di output inoltre la shell crea anche un processo per wc -l e questa è la quarta linea
ps | tee t| wc -l qua avrei 5 perché si è aggiunto anche il processo per eseguire tee

kill+ PID del processo: per uccidere un processo dell'utente in background
&*:* per compiere un processo in background
ls -lR / >temp 2> temperror & processo lunghissimo che mando in background con &
nella prima riga tra le quadre ho il numero di processi attivi in background
grep pippo < temp | more cerca in temp pippo e fai una visualizzazione paginata

date: riporta data e orario odierno (CET: central europeal time)
diff+file1+file2: confronta i due file , ci mostra le linee diverse e quelle in più

find + directory+ -name nomefile: cerco in una certa directory un certo file
find . -name *pippo* cerca nella directory nei file la stringa pippo

COME ESEGUIRE UN FILE COMANDI
chmod +x F.sh: da diritto di esecuzione al file
./F.sh: lancio il file
oppure
sh F.sh: invoco direttamente la shell passandogli il file

sh -x	stampa i comandi e gli argomenti come sono stati eseguiti
sh -v	stampa le linee del file comandi come sono lette dalla shell

#: commento fino alla fine della riga

QUANDO CREO UN FILE
all'inizio di ogni file comando **#!/bin/sh**
per renderlo eseguibile **chmod +x nomefile**
ls -l + nomefile se è verde ok
eseguo con **./nomefile.sh**

$ + nomevariabile: per recuperare il valore della variabile
a=10
echo $a stampa il valore 10
nomevariabile=valore: assegnamento, NON CI SONO SPAZI NE PRIMA NE DOPO L'UGUALE

env: (environment) mostra l'ambiente corrente di ogni processo shell

echo $PATH: mi mostra i cammini contenuti in path separati da i due punti
echo $HOME: ci dice qual è la nostra home

PER SCRIVERE DUE COMANDI LI SEPARO DAL PUNTO E VIRGOLA
echo $PATH; echo $HOME

export + nomevariabile: per far diventare variabile d'ambiente(variabili ereditate) una variabile di shell(variabili proprie del proc)

unset + nomevariabile: per cancellare una variabile sia di shell che di ambiente

SOSTITUZIONI

1. di variabili/parametri
   a=p con p file testo
   echo $a stampa p
   ls -l $a mi mostra le informazioni di p(è uguale a ls -l p)

2. di comandi
   echo `pwd` /$a visualizza in st. output il nome assoluto del file p, utilizzato quindi per mostrare il nome per intero
   mentre usiamo i comandi
   `comando`: espande un comando, cioe viene eseguito e sostituito con il suo output
   echo `pwd`/F stampa il nome completo del file F

3. dei nomi di file
   *ls -l `pwd` /$a** ci mostra il comando ls -l applicato a tutti i nomi

assoluti dei file che iniziano per p nella directory corrente

echo [abc]*	stampa i nomi di tutti i file che iniziano per a, b e c
echo [q-s]*	stampa i nomi di tutti i file che iniziano con lettere che nell'alfabeto sono comprese tra q e s
echo *[0-9]	stampa i nomi di tutti i file che finiscono con cifre comprese tra 0 e 9
echo *[!0-9]	stampa i nomi di tutti i file che non finiscono con cifre comprese tra 0 e 9
ls *[!0-9]	stampa prima tutti i nomi di file che rappresentano la specifica poi le directory e il loro contenuto
echo [a-p, 1-7]*[o, a]	stampa tutti i file la cui iniziale sia un carattere tra a e p o una cifra da 1 a 7 e contemporaneamente deve finire per o oppure a
echo *	stampa tutti i nomi
echo *	stampa tutti i nomi che iniziano con asterisco

`expr $nomefile operazione valore`: consente di fare operazioni aritmetiche con i valori delle variabili (che rimangono stringhe), *j=$i+1* *questa operazione semplicemente aggiunge alla stringa 'i' i caratteri '+1'*
j = `expr $i +1` sommo uno al valore in i e lo metto in j
N.B: per la moltiplicazione con l'asterisco devo usare l'escape se no me lo guarda come metacarattere
j = `expr $i \* 2`

comando + '…' e "…" con dentro le istruzioni: per inibire le sostituzioni
y=3
*echo $y e `pwd` e ** $y prima sostituzione, `pwd` seconda sostituzione, * terza sostituzione
*echo '$y e `pwd` e *'* non viene svolta nessuna sostituzione
*echo "$y e `pwd` e *"* vengono eseguite solo le prime due sostituz.
PER ESEGUIRE IL CONTENUTO DEL FILE SCRIVO **nomefile.sh**
la prima riga stampa
3 e percorso directory del file *e* tutti i file (perché ho scritto solo *)
la seconda riga stampa
*$y e `pwd` e **
la terza riga stampa
3 e percorso directory del file *e **

x= 'ls -l $z' a x assegno la stringa scritta pari pari (non avvengono sostituzioni)
z=provaSost.sh assegno a z il contenuto del file
$x eseguo x quindi viene fatto ls -l ma $z rimane così perché viene attuata solo la prima sostituzione (mi darà errore)
eval $x in questo caso eseguo anche il $z e quindi z ha il valore di provaSost.sh
sh -x provaSost.sh verifico che sia stato eseguito tutto
comando argomento1 argomento2: passo al file comandi dei parametri
esempio:
provaPar.sh ciao 5 ecco qui eseguo il file e gli passo questi 4 parametri
posso passare come parametri anche dei file
DIR.sh ttt prende come parametro il file ttt
oppure una directory
DIR.sh d1 prende come parametro la directory d1
se non metto parametri non verrà stampato nulla al posto di $1, se voglio fare per es
ls -l $1 lo farà della dir corrente
DIR. SH | more per paginare un output

$1 $2: richiamo nel mio file i parametri passati come primo e come secondo
$0: mi dice il nome del comando che ho usato per i parametri
shift: assegna ad ogni parametro il valore del parametro successivo
(non ho effetto su $0, il valore di $1 si perde)
set+valore1+valore2: assegna ai parametri nuovi valori (al primo valore1, al seconde valore2 ecc)
$*: mostra i valori di tutti i parametri
$#: il numero di parametri (senza contare il comando)
$?: valore di ritorno dell'ultimo comando eseguito
$$: il PID del processo in esecuzione
CICLO IF
If lista comandi
then comandi
else comandi
fi
test +nomefile: valuta l'espressione, 0 se ha successo diverso se ha insuccesso

test -f	verifica l'esistenza del file specificato nella directory corrente, se no devo specificare il percorso
test -d	verifica l'esistenza della dir specificata nella directory corrente, se no devo specificare il percorso
test -r	verifica se abbiamo diritto di lettura sul file/dir specificato
test -w/-x	come prima ma di scrittura o esecuzione
test str1 = str2	valuta se due str sono uguali (DEVO METTERE GLI SPAZI)
test str1 != str2	valuta se due str non sono uguali (DEVO METTERE GLI SPAZI)
test -z str1	valuta se la str è nulla
test str1	valuta se str non è nulla
test num1 [-eq, -ne, -gt, -ge, -lt, -le] num2	confronta due stringhe numeriche con i vari operatori relazionali

posso anche comporle con **-a**(AND) **-o**(OR) e **!**(NOT)
per controllare di passare il numero giusto di parametri è controllarlo con un if

```
if test $# -ne 2
then echo $# non numero giusto di parametri!
        exit 1
fi
if test -f $2
then grep $1 $2 > /dev/null 2> &1
if test $? -eq 0
then echo TROVATO STRINGA $1 NEL FILE $"
else echo NON TROVATO STRINGA $! NEL FILE $"
fi
else echo il file $2 non esiste
fi
```

LA DIFF TRA = e -eq
esempio

```
a=1
b=01
if test $a -eq $b; thrmen echo OK; else echo NO; fi
```

stampa OK

```
if test $a = $b; then echo OK; else echo NO; fi
```

stampa NO

echo > /dev/tty: mi assicura che ciò che stampo finisca sullo std output
read+var1: ci fa leggere dati dallo std input, quello che c'è in standard input verrà messo in var 1
esempio
echo "vuoi visualizzare il file $1 (si/no)?" > /dev/tty la mia stringa sarà stampata in output
read var1 la risposta dell'utente sarà letta e messa in var1
COSTRUTTO CASE
case $var **in**
pattern1**)** comandi**;;**
….
pattern i | pattern j | pattern k**)** comandi**;;**
esac
questo costrutto è un po' più flessibile dell'if
esempio:
echo "Fornire una risposta (affermativa ===> Si, si, Yes, yes)"
read risposta
case $risposta in
S | s* | Y* | y*) echo OK;;*
**) echo NO;;*
esac
COSTRUTTO FOR
for var [**in** list] in questo caso var è il nome della variabile che verrà usato per indicare le varie iterazioni del for
do
comandi
done
esempi:

1. *for i in **
   do
   echo
   $i
   done

questo ciclo stampa tutti i nomi di file

2. se mettessi for i in p* stamperebbe quelli con la p
3. *for file in `cat ttt`; do echo $file; done*

questo for stampa solo il nome di tutti i file contenuti in ttt
(*ls -l `cat ttt`* dove ttt è un file che contiene dei nome di file,
questa riga stampa i nomi di file e le loro info)
FOR NON SPECIFICATO
for i prende come lista la lista dei parametri (**in** $*)
do > $i
done
esempio: in crea ho scritto
for i lista degenere (contiene solo un elemento)
do
>$i ridirezione a vuoto
done
../crea. sh f t a r t file1 file 2 crea in una dir vuota tutti questi file
CICLO WHILE
while lista-comandi
do
comandi
done
si ripete finché l'ultimo comando ritorna 0, cioe ha successo

esempio:
while test ! -f $1
do
sleep 10
done
ogni volta che il file passato non esiste si sospende per 10 secondi poi rifà la ricerca, questo può essere utile per tenere in attesa il processo aprire un'altra finestra creare il file e tornare alla prima.
sleep + numero: sospende il processo all'interno del quale è contenuto per tanti secondi quanti specificati dal numero
CICLO UNTIL
until lista-comandi
do
comandi
done
si ripete finché l'ultimo comando ritorna 1, cioe non ha successo
esempio:
until who | grep $1
do
sleep 10
done
Per vedere se un utente passato come parametro è collegato