- Trovare i file con nome dispari e stamapre il nome del file e il numero di caratteri (viene un carattere in più perchè conta lo \n)

```
for file in $(find . -name '*.txt');
do n=$(echo $file | awk -F/ '{print $NF}' | awk -F'.' '{print $1}' | wc -m);
if [[ $n%2 -ne 0 ]]; then echo $file " " $n; fi; done
```

- Trovare i file...Versione corretta

```
for file in $(find . -name '*.txt' | awk -F/ '{print $NF}' | awk -F'.' '{print $1}');
do n=$(echo ${#file}); if [[ $n%2 -ne 0 ]]; then echo $file " " $n;fi; done
```

- Trovare i file .c o .h con nome dispari di dimensione minore di 3k e stampare quanti sono

```
for file in $(find / -type f -name '*.[ch]' -size -3k 2> /dev/null | awk -F/ '{print $NF}' | awk -F.'
'{print $1}');
do n=$(echo ${#file}); if [[ $n%2 -ne 0 ]]; then echo "$file"; fi; done 2> /dev/null | wc -l
```

- Per ogni utente stampare la somma di dimensione dei file (fatto con xargs du)

```
for user in $(awk -F':' '{print $1}' /etc/passwd);
do size=$(find / -user $user -type f 2> /dev/null | xargs du -bc | tail -1 | awk '{print $1}');
echo "$user $size"; done
```

- Per ogni utente stampare la somma di dimensione dei file (fatto con printf)

```
for target in $(awk -F':' '{ print $1}' /etc/passwd);
do res=$(find / -user $target -type f -printf '%p|%s\n' 2>/dev/null |  awk -F\| '{ s += $2 } END {if(s!
=0){print "Sum: ",s}else{print "Sum: 0"}}');
[[ $res != "" ]] && echo "$target $res"; done | sort -k3 -n -r | head -n 3
```

- Estrarre tutti gli indirizzi IP contenuti nei soli file regolari presenti nel ramo /etc del filesystem

IPV4

```
grep -r -E "((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9])" /etc
```

Senza validazione

```
find /etc -type f -exec grep -osE "(([0-9]){1,3}\.){3}([0-9]{1,3})" {} \; 2>/dev/null
```

Con Validazione

```
find /etc -type f -exec grep -osE "((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|
[01]?[0-9])" {} \; 2>/dev/null
```

IPV6

```
grep -oE "([0-9a-f]{0,4}:){0,7}[0-9a-f]{0,4}" /etc
grep -oE "([0-9a-f]{0,4}:){3,7}[0-9a-f]{0,4}" /etc

find /etc -type f -exec grep -osE "([0-9a-f]{0,4}:){0,7}[0-9a-f]{0,4}" {} \; 2>/dev/null
```

- 1)Conta il numero di include e define (maiuscoli e minuscoli) nei file .h o .c. nel sistema
grep il -i serve per ignorare la differenza tra maiscolo e minuscolo, - o stampa solo il match esatto che trova, se non lo mettessi mi metterebbe tutta la riga nella quale trovo il match

```
find / -type f -name "*.h" -o -name "*.c" 2>/dev/null| xargs cat| grep -i -o 'define\|include'| tr '[:upper:]' '[:lower:]' | sort | uniq -c
```

- 1.1) pipe che cerca i file txt e stampa il nome di file seguito dalle righe pari in essi contenute, awk in quetso caso lo uso per fare il controllo di parità delle righe

```
for file in $(find /home -type f -name "*.txt"); do echo $file ; cat $file | awk 'NR%2==0'; done
```

- 2)#Calcolare lo spazio occupato dai file con un numero di righe pari
Inizializzo s=0, scorro tutti file txt , stampo il nome del file seguito dal conteggio della linea (wc -l) e lo mette sulla stessa linea dato che uso printf. Finisce il for seleziona solo le righe che hanno il secondo elemento pari, e ne stampa il primo elemento (che è il nome del file), du metet accanto al nome la dimensione, selezionamo e stampiamo solo la dimensione e usiamo un' ultima volta awk per fare la somma sulla variabile s.

```
s=0; for file in $(find /home -type f -name "*.txt"); do printf $file" "; cat $file | wc -l  ; done | awk '$2%2==0' | awk '{print $1}'| xargs du | awk '{print $1}' |awk  '{s=s+$1} END {print s}'
```

- 3)#trovare tutti i file modificati di venerdì.

%p mi stampa il percorso del file trovato e ta la data dell'ultimo accesso. Con Grep vado a cercare (mi seleziona quelli con la parola Fri)

```
find / -type f -printf '%p %ta \n' 2>/dev/null | grep -E Fri | awk '{print $1}'
```

- 4)# Per ogni utente del sitema conti il numero di file posseduti e li ordini in maniera decrescente
awk lo uso per dividere sui due punti e stampo il primo parametro (non a video ma è il valore che faccio assumere alla variabile del for user), poi stampo la variabile user (il nome di ogni utente) , trovo tutti i file che hanno il propietario uguale allo user che sto considerando e li conto con wc -l (lo faccio per ogni user del ciclo), termino il for, e uso awk prima per dividere la stringa in input sullo spazio (il contenuto delle virgolette) e poi ristampo la striga con i parametri invertiti in modo tale da poterli ordinare con il sort (e nella terza riordino anche in modo da avere prima l'utenet e solo dopo il nuemro)

```
for user in $(find /etc/passwd | xargs cat |awk -F':' '{print $1}'); do printf $user" "; find / -type f -user $user 2>/dev/null | wc -l ; done | awk -F' ' '{print $2" " $1}'| sort -n
```

```
for user in $(find /etc/passwd | xargs awk -F':' '{print $1}'); do printf $user" "; find / -type f -user $user 2>/dev/null | wc -l ; done | awk -F' ' '{print $2" " $1}'| sort -n
```

```
for user in $(find /etc/passwd | xargs awk -F':' '{print $1}'); do printf $user" "; find / -type f -user $user 2>/dev/null | wc -l ; done | awk -F' ' '{print $2" " $1}'| sort -n | awk -F' ' '{print $2" " $1}'
```

- 5) Trovare le parole nelle righe pari e queste parole devono avere il pattern vocale-consontante.vocale (precedente) (sono riuscito a farlo solo se il pattern è alla fine della parola)

cerco tutti i file, li ridirigo in standard output con cat, prendo le righe pari, prendo le parole con il primo grep (la o fa sì che io prenda solo le parole) , con l'altro grep richiedo la sequenza vocale consonante vocale.

```
for file in $( find /home/giona/Desktop/ -type f 2>/dev/null ); do cat $file |  awk 'NR%2==0' |  grep -ioE '[[:alpha:]]+' | grep -iE '(.*)([aeiou])([^aeiou])\2\1' ; done  2>/dev/null
```

- 6)Stampare i 3 file più grandi per ogni utente

```
for user in $(find /etc/passwd |xargs cat| awk -F ':' '{print $1}'); do echo $user; find / -type f -user $user 2>/dev/null |xargs  du 2>/dev/null| sort -n | tail -3 ; done
```

- 7)Copiare i file di giona e che iniziano per vocale da una cartella all'altra, mantenendo inalterati i permessi
Versione 1 che per controlalre che inizi con vocale il nome del file impone che prima del nome del file ci debba essere tutto il percorso

```
mkdir landing; find /home/giona/Desktop/lab/ -type f -user giona 2>/dev/null| grep -iE '(/home/giona/Desktop/lab/)([aeiou])(.*)'| xargs -I var cp --preserve var /home/giona/Desktop/landing/
```

Con questa invece riesco ad eliminare il dover mettere il percorso, ma poi non funziona la copia dato che ho rimodellato i file e non sono più un percorso completo...

```
find /home/giona/Desktop/lab/ -type f -user giona 2>/dev/null|awk -F '/' '{print $6}' | grep -iE '(^[aeiou])(.*)'
```

- 8)Pipe che usa sed, molto utile per sostituire delle parole, in questo caso sostituisce copia (seguito da uno spazio con nulla), in sed la -e (expression) e la g

```
find /home/francesco/Scrivania/EserciziSisop -user francesco -type f -name "*.txt" 2>/dev/null | awk -F'/' '{print $6" "$7}' | sed 's/copia //'
```

- 9)Stampare la dimensione ed i file con dimensione compresta tra 8MB e 2GB
il bh alla fine fa sì che sua human-reable

```
find /home/giona/Desktop/lab -type f -size +8M -size -2G |xargs du -bh
```

- 10)Comprimere tutti i file di dimensione compresa tra 2MB e 2GB
c=create z=formato zip

```
find /home/giona/Desktop/lab -type f -size +2M -size -2G |xargs tar -cvzf compresso.tar
```

- 11)Pipeline che legge il file testo.txt nel percordo nel quale ci troviamo e legge le parole, le ordina, le conteggia, le riordina per apparizione in maniera decrescente

```
pwd | xargs -I var find var/testo.txt | xargs -I var cat var | grep -ioE '[[:alpha:]]+' | sort | uniq -c | sort -nr
```

- 12)Pipeline che cercano i file che contengono la parola casa e stampa il nome dei file che soffisfano la condizione

```
find /Users/giona/Desktop -type f -name '*.txt' |xargs grep -iE 'casa'
```

oppure con il for (ma non stampa il contenuto del file)

```
for file in $( find /home/francesco/Scrivania/EserciziSisop -type f -printf "%p \n"); do echo $file | xargs grep -ie 'ciao' -l; done
```

- 13)Trova tutit i file png e li cambia in jpg, fino a prima di bash creo delle strinhe che sono come un comando che fa move del primo elemento nel secondo
find /home/giona/ -name '*.png' | sed -e 's/.*/mv & &/' -e 's/png$/jpg/'| bash

ESERCIZI TIPICI

Specificare l'occupazione totale di un utente in un sistema (somma delle dimensioni dei suoi file...)

```
#!/bin/sh
if [ $# -ne 1 ]; then
  echo "syntax: user_size <user_name>"
  exit 1
fi
ID="$(grep "^$1" /etc/passwd | cut -f 3 -d :)"
if [ "$ID" = "" ]; then
  echo "User \"${1}\" not found"
  exit 1
fi
echo "calculating..."
SUM=0
for ITEM in $(find / -user "$ID" -type f | xargs du | cut -f 1); do
  SUM=$(expr ${SUM} + ${ITEM})
done
echo "User \"${1}\" takes ${SUM} KB into the system"
exit 0
```

Per ogni utente del sistema specificare qual è il suo file di dimensioni maggiori

```
#!/bin/sh
# Per ogni utente trovare il suo file di dimensione maggiore
for UID in $(cut /etc/passwd -d : -f 3 | sort -n | uniq); do
  FILE=$(find / -type f -user $UID | xargs du | sort -nr | head -1)
  if [ "$FILE" != "" ]; then
    DIM=$(echo "$FILE" | cut -f 1)
    NOME=$(echo "$FILE" | cut -f 2)
    echo "UID: $UID => File: $NOME ($DIM KB)"
  else
    echo "UID: $UID => Nessun file trovato"
  fi
done
```

Trovare il file più "grosso" sul sistema

```
find / -type f | xargs du | sort -nr | head -1
find / -type f | xargs du | sort -n | tail -1
```

Archiviare i files più grandi di 5k e più piccoli di 100k?

```
find / -type f -size +10 -a -size -200 | xargs tar cvf archivio.tar
```

Copiare i file da una directory all'altra senza cambiare i permessi:
cp /directory/albero1/* /directory/albero2

Spostare tutti i files che iniziano per m da una directory (e sottodirectory) ad un'altra
mv /directory/albero1/m* /directory/albero2

Trovare tutti i files eseguibili > 5k e evidenziando i 5 più grandi
find / -type f -size +10 | xargs ls -lF | grep "*$" | tr -s ' ' ' ' | cut -d ' ' -f 5,9 | sort -n | tail -5

Archiviare i files modificati nell'ultima settimana
find / -type f -mtime -7 | xargs tar cvf archivio.tar

Trovare il processo che occcupa più memoria ram
ps -axl | sort +7 -nr | head -1 | tr -s ' ' ' ' | cut -d ' ' -f 8,12

Trovare i tre files più grossi per ogni utente

```
#!/bin/sh
for UID in $(cut /etc/passwd -d : -f 3 | sort -n | uniq); do
  echo "Utente $UID:"
  find / -type f -user $UID | xargs du | sort -nr | head -3
done
```

```
#/usr/bin/sh
cut -f 1 -d : /etc/passwd |while read user
do
   file=$(find / -type f -user $user |xargs du |sort -nr |head -1|cut -f 2)
   dimensione=$(find / -type f -user $user |xargs du |sort -nr |head -1|cut -f 1)
   if [ "$file" != "" ];then
      echo "il file " $file " di dimensione "$dimensione " e'il piu' grosso per l'utente "$user
   else
      echo "l' utente " $user "non ha file"
   fi
```

done

Archiviare tutti i files che contengono la parola "login"

find / -type f | xargs grep -l "login" | xargs tar -cvf archivio.tar

---

Trovare tutti i file che includono la parola "login" ed archiviarli su di un file .txt

find / -type f | xargs grep -l "login" >> list.txt

Cercare tutti i file con SUID attivato
find / -type f -perm -004000 (testato)

Archiviare i files più piccoli di 100KB ma più grandi di 5KB:

find / -type f -size +10 -a -size -200 | xargs ar rc archive.a

Opzioni:
r => sostituisci o aggiungi (nel caso un file non sia già presente)
c => crea l'archivio (sopprime il noioso messaggio di notifica, non fa nulla se l'archivio esiste già)
t => visualizza il contenuto (ar t archive.a)

Trovare tutti i processi per ogni utente e fornirne l'occupazione in memoria???

```
#!/bin/sh
SUM=0
for UID in $(cat /etc/passwd | cut -d : -f 3); do
  for PROCSIZE in $(ps alx | tr -s ' ' ' ' | cut -d ' ' -f 4,8 | grep "^$UID" | cut -d ' ' -f 2); do
    SUM=$(expr $SUM + $PROCSIZE)
  done
  echo "Utente $UID : $SUM KB di processi in memoria"
  SUM=0
done
```

ps alx => stampa tutti i processi in formato lungo
tr -s ' ' ' ' => comprimo tutti gli spazi contigui in un solo spazio (fondamentale, altrimenti non funziona il cut)

cut -d ' ' -f 4,8 => estrai la 4^ e l'8^ colonna (che dopo tr sono separate da un solo spazio) che sono l'UID del proprietario e la dimensione (le ho trovate osservando il comportamento dei comandi precedenti, non c'è un manuale su come farlo!)
grep "^$UID" => cerca l'UID interessato a partire dall'inizio della riga (potrebbe confondersi con la dimensione di un processo)
Ricordo che le colonne ora avranno il formato "0 689" ovvero UID + spazio + SIZE
cut -d ' ' -f 2=> Una volta ristretto il campo dell'utente estraggo la sola colonna dei SIZE per poi sommarli tutti

---

Trovare il processo con pid dispari che occupa più spazio in memoria?

```
#!/bin/sh
MAXSIZE=0
MAXPID=0
for PROC in $(ps alx | tr -s ' ' : | cut -d : -f 5,8); do
  PID=$(echo $PROC | cut -d : -f 1)
  SIZE=$(echo $PROC | cut -d : -f 2)
  if [ $PID % 2 -eq 1 -a $SIZE -gt $MAXSIZE ]; then
    MAXPID=$PID
    MAXSIZE=$SIZE
  fi
done
echo "Processo con PID dispari più esoso: $MAXPID con $MAXSIZE KB"
```

Spiegazione:
Dalla lista di tutti i processi comprimo tutti gli spazi contigui nel carattere ':'
che userò come separatore di colonne (lo spazio faceva danno); estraggo le colonne 5 e 8 che sono PID e SIZE
anche qui trovate osservando il comportamento dei comandi)
Le righe ora sono della forma PID:SIZE (12:552)

Estraggo i PID come prima colonna dalla variabile PROC
Estraggo i SIZE come seconda colonna dalla variabile PROC
Se il PID % 2 è uguale a 1 (dispari) e SIZE è maggiore di MAXSIZE
Aggiorno il PID e il SIZE massimi

Calcolare la somma della dimensione di file eseguibili che hanno all'interno la parola copyright non sensitive

```
#! /bin/sh

SUM=0
for SIZEFILE in $(find / -type f | xargs grep -il "copyright" | xargs ls -lF | grep "\*$" | tr -s " " : | cut -d : -f 5);
do
SUM=$(expr $SUM + $SIZEFILE)
done
```

```
echo "la somma è: $SUM"
```

le opzioni -il del grep mi consentono di:
-i per il case insensitive
-l per elencare il nome del file contente la parola copyright
(altrimenti mi avrebbe stampato per ogni file la riga contenente la parola copyright)

Trova il processo che occupa per ogni utente più memoria

```
#! /bin/sh

MAXSIZE=0

for USER in $(cat /etc/passwd | cut -d : -f 3 | uniq); do

for PROC in $(ps -axl | tr -s " " : | cut -d : -f 4,8 | grep -v "SZ"); do
UID=$(echo $PROC | cut -d : -f 1)
SIZE=$(echo $PROC | cut -d : -f 2)

if [ $UID -eq $USER -a $SIZE -gt $MAXSIZE ]; then
MAXPROC=$PROC
MAXSIZE=$SIZE
fi

done
echo "il processo dell'utente $USER + grande è: $MAXPROC"
MAXPROC="nessun processo"
MAXSIZE=0
done
```

Copiare una directory mantenendo la struttura delle sottodirectory e i permessi, evitando però di copiare il contenuto delle cartelle(i file che all'interno che non sono directory)

```
#/bin/sh
cpdir -p /directoryA /directoryB
for file in $(find /directoryB -type f); do
rm $file
done
```

oppure

```
#! /bin/sh
cpdir -pr /bin /bin2
rm $(find /bin2 -type f)
```

Quali e quante estensioni ci sono nel sistema?
(per estensioni si intende qualsiasi cosa ci sia dopo il carattere “.”, es: archivio.gz)

La soluzione è questa:

find / | rev | cut -f 1 -d '.' | grep -v / | rev | sort | uniq -c | sort -n

Spiegazione:

find / trova tutto (files e directories) a partire dalla root dir
rev inverte ogni riga dell'output
cut -f 1 -d '.' estrae il primo campo di ogni riga usando il . come separatore
grep -v / eliminazione delle sole directories
rev inverte di nuovo ogni riga dell'output
sort ordina l'output
uniq -c elimina righe duplicate contandole, ora l'output è del tipo 123 gz
sort -n (non necessario) ordina i risultati numericamente

Nota:
Si può evitare il grep -v / usando cut -f 1 -d '.' -s.

Okkio che così trovi anche tutte le cartelle e file nascosti (del tipo .cartella o .file).
Ho fatto uno script simile al tuo, escludendo cartelle e file nascosti:

find / -type f -name *.* | tr -s '/' ' ' | rev | cut -d ' ' -f 1 | rev | grep -v "^\." | rev | cut -d '.' -f 1 -s | rev | sort | uniq -c

find / -type f -name *.* cerco tutti i file che contengono un punto nel nome del file (i file trovati sono completi di path ad es. /root/prova/file)
tr -s '/' ' ' spazi al posto di /
rev stampo il nome del file al contrario
cut -d ' ' -f 1 estraggo il primo campo (è il nome del file al contrario). In questo modo ho il nome del file senza path
rev ristampo il nome del file "dritto"
grep -v "^\." escludo tutti i file che INIZIANO con un punto (sono quelli nascosti)
rev non lo ripeto più
cut -d '.' -f 1 -s estraggo il primo campo. Così ho l'estensione del file
rev
sort ordino le estensioni in ordine alfabetico, così posso eliminare le ripetizioni con uniq
uniq -c per ogni estensione ho il numero di ripetizioni

Qualcuno sa cosa significa fare la statistica dei file < 10k 100k 1000k ??????????

```
#! /bin/sh
TOT=$(find / -type f | wc -l | tr -s ' ' : | cut -f 2 -d :)
DIECIK=$(find / -type f -size -20 | wc -l | tr -s ' ' : | cut -f 2 -d :)
CENTOK=$(find / -type f -size -200 | wc -l | tr -s ' ' : | cut -f 2 -d :)
MILLEK=$(find / -type f -size -2000 | wc -l | tr -s ' ' : | cut -f 2 -d :)
```

```
CENTO=100
STAT=$(expr $DIECIK \* $CENTO)
STAT1=$(expr $STAT / $TOT)
STAT=$(expr $CENTOK \* $CENTO)
STAT2=$(expr $STAT / $TOT)
STAT=$(expr $MILLEK \* $CENTO)
STAT3=$(expr $STAT / $TOT)
echo "I file inferiori a 10k sono il ${STAT1}%"
echo "I file inferiori a 100k sono il ${STAT2}%"
echo "I file inferiori a 1000k sono il ${STAT3}%"
```

Per trovare ogni singolo file di testo che contiene la parola "copyright" basta fare:

```
find / -type f | xargs grep -l "copyright" | grep "\.txt$"
```

Somma delle dimensione di tutti i file di solo testo (non eseguibili) che contengono al loro interno la parola copyright

```
#!/bin/sh

SOMMA=0
for VAR in $(find / -type f | xargs grep -l "copyright" | grep "\.txt$" | xargs du | cut -f 1); do
SOMMA=$(expr $VAR + $SOMMA)
done
echo $SOMMA
```

---

per cercare un eseguibile (però è stato detto più volte) fai così

```
find / -type f | xargs ls -lF | tr -s ' ' ' ' | cut -d ' ' -f 5,9 | grep "\*$"
```

dove ls -F mette in fondo al nome del file alcuni simboli... mette un * se è eseguibile
qunidi con tr ' ' ' ' comprimo tutti gli spazi
con cut, con delimitatore lo spazio (-d ' ') selezioni i csmapi 5 e 9 (PID e SIZE)
e poi estraggo le linee che hanno * alla fine (devo mette \ davanti a * per farlo interpretare bene...
e il $ significa "cerca in fondo alla riga"

Trovare tutte le dir che abbiano meno di 5 sottodirs:

```
#!/bin/sh
echo "Script per stampare le dir che hanno meno di 5 sottodir";
```

```
for dir in $(find / -type d);
 do
      ls -F -1 $dir | grep '/' | wc -w > a.tmp
      if [ $(cat a.tmp) -lt 6 ];
         then
            echo $dir;
            cat a.tmp;
         else
            echo '';
       fi;
      rm a.tmp;
done;
```

Trovare il file più grande di tipo testo che abbia un numero di righe pari

```
sizef=0
for file in $(find / -type f -name '*.txt'); do
 nrrighe=$(wc -l $file | tr -d ' ' | cut -f 1 -d /)
 size=$(du $file | cut -f 1)
 if $(expr $(expr $nrrighe % 2) -eq 0 ); then
    if $(expr $size -gt $sizef); then
       sizef=$size
       filef=$file
    fi
 fi
done
echo "Il file di testo più grande con numero di righe pari è:
echo $filef
```

Calcolare per ogni utente il numero di file modificati nell'ultimo mese

```
for user in $(cut /etc/passwd -d : -f 3 | sort -n | uniq); do
item=$(find / -type f -user $user -mtime -31 | wc -l | tr -d ' ' | cut -f 1)
echo "L'utente $user ha modificato $item file nell'ultimo mese"
done
```

Per ogni utente del sistema stampare i gruppi a cui appartiene, senza utilizzare i comandi id e groups.

```
for x in $(cat /etc/passwd |cut -d ':' -f 1); do
   echo "$x: ""$(for y in $(cat /etc/group|grep -s $(cat /etc/passwd|grep -s "$x"|cut -d ':' -f 3)|cut -d ':' -f 1);
do
      echo -n "$y ";done)";
```

```
done
```

Trovare tutti gli script del sistema, minori di 1 k, copiarli su /tmp/script/ Aggiungerli tutti i file su un file.tar !

```
#!/bin/sh
mkdir script;
FILE=$(find / type -f -size +2 | xargs grep -l "#!/bin/sh" | cp * /tmp/script);
```

Trovare tutti i file modificati di venerdì

```
#!/bin/sh

find / -type f |while read line
do
giorno=$(stat -Mtime $line |awk '{$1 print $1}')
if [ "$giorno" = "Fri" ]; then
   ls -l $line
fi
done
```

Memorizzare in un archivio .ar tutti i file creati negli ultimi 2 giorni di dimensione minore di 5k.

```
find / -type f -size -10  -ctime -2 |xargs ar rc  filearc.ar
```

Calcolare per ogni utente il numero di file modificati nell'ultimo mese

```
cut  -f 1 -d : /etc/passwd |while read line
do
  conta=$(find / -type f -mtime -30  -user $line|xargs ls -l|wc -l)
  echo "lo user " $line "ha modificato " $conta "file negli ultimi 30 giorni "
done
```

Calcolare la dimensione totale di tutti i file .c

```
#/usr/bin/sh
```

```
find / -type f -name *.c |
while read line
do
  conta=$(stat -size $line)
  tot=$(expr $conta + $tot)
  echo $tot > totale
done
```

Memorizzare in un archivio .ar tutti i file creati negli ultimi 2 giorni di dimensione minore di 5k.

```
find / -type f -size -10  -ctime -2 |xargs ar rc  filearc.ar
```

Per ogni utente trovare i 3 file più vecchi del sistema..

```
#!\bin\sh
USERS=$(cut -d : -f 1 | sort -f | uniq);
for us in $USERS; do
        FILE=$(find / -type f -user $us | xargs ls -ltsT | tail -3);
        echo "FILE UTENTE: $us;
        echo $FILE;
done
```

Calcolare il numero di righe totali che ha scritto sul filesystem un certo utente nell'ultimo mese

```
find / -mtime -30 |while read line ;
do;
  awk '{ if ($(ls -l $linea)) $3 == 'root' print }';
done;
```

```
find / -mtime -30 |while read line ;
do;
ls -l $line | awk '{ print $3": "$9}' | grep ^root
done;
```

Calcolare la somma delle dimensioni dei file *.s.

```
#/usr/bin/sh


find / -name *.s | while read line;
```

```
do;
                sum=$(ls -l $line| awk '{print $5}');
          ####     pippo= $(ls -l $line| awk '{sum = sum + $5} END {print sum}'
                        ###echo $sum;
tot=$(expr $sum + $tot);
      echo $tot > tmp;
                                done;
```

Trovare tutti i file che hanno il numero di blocchi pari

```
#/usr/bin/sh
###sum=0;
find / |while read line;
do;
ls -l $line| awk '{if ($5%2==0) print }';
done;
```

Trovare i file più vecchi di un mese

```
find / -mtime -30 |while read line ;
 do;
          awk '{ if ($(ls -l $linea)) $3 == 'root' print }';
 done;
```

Mostrare, per ogni utente, il numero di file presenti sul sistema.

```
find / -f |xargs wc -l
```

-Calcolare il numero di righe totali che ha scritto sul filesystem un certo utente nell'ultimo mese

---

Stampare i file che finiscono per c con un numero di caratteri dispari e visualizzare la loro data

```
#/usr/bin/sh

find / -name *.c |while read line ;
do;
      wc -c $line|awk '{if ($1%2 != 0 ) ls $2 print $2}'|xargs ls -l;
done;
```

Trovare tutti i file col numero dispari di righe (quindi sono i file di testo)

find / -f|while read line ; do wc -l $line| awk '{if (($1%2) !=0) print $line}' ;done;

Trovare tutti i file modificati di venerdi' (sia un venerdi' particolare che di venerdi' in generale)

```
#/usr/bin/ksh
find / -type f|while read line;
 do ; stat $line |awk '{if ($1 =="Mtime:" && $2 =="Fri") print "'${line}'" }' ;
 done;
```

Trovare tutti i link simbolici presenti nel sistema.
find / | while read line;do if test -h $line; then echo $line ok; fi; done

---

Trovare tutti i file col numero dispari di righe (quindi sono i file di testo)
find / -f|while read line ; do wc -l $line| awk '{if (($1%2) !=0) print $line}' ;done;

Trovare gli utenti ed i gruppi distinti proprietari dei file nelle directory /bin, /sbin, /usr/bin e /usr/sbin.
Riportare i comandi utilizzati.
ls -l /bin /sbin /usr/bin /usr/sbin |awk '{print $3, $4}'|sort |uniq

---

PARTIZIONE

```
df -h
sudo fdisk /dev/discoNonMontato
roba su fdisk
mkdir mount
sudo mkfs.ext2 /dev/partizioneDaMontare
sudo mount /dev/partizioneDaMontare ./mount
dd if=/dev/zero of=/home/part/mount/file bs=nBytes count=1
```

PIPE

contare l'occorrenza di parole all'interno di un file senza considerare la punteggiatura e metterli in ordine decrescente (più frequenti in alto)

```
grep -oE '[[:alpha:]]+' testfile.txt | sort | uniq -c | sort -nr
```

prendere con l'occorrenza maggiore all'interno di un file

```
grep -oE '[[:alpha:]]+' testfile.txt | sort | uniq -c | sort -nr | head -n 1
```

stessa domanda sopra senza usare head

```
grep -oE '[[:alpha:]]+' testfile.txt | sort | uniq -c | sort -n | tail -n 1
```

PER NON FARE USCIRE ERRORI DA UN BLOCCO (attenzione che toglie errori solo di quel blocco della pipe, in questo caso di find) find .... 2>/dev/null

stampa parole con occorrenza più alta di file che iniziano con s e hanno estensione .h che si trovano in include

```
find /usr/include/ -name 's*.h' 2>/dev/null | xargs -Ivar grep -oE '[[:alpha:]]+'
var | sort | uniq -c | sort -nr | head
```

stampa parole con occorrenza più alta di file che iniziano con s e hanno estensione .h che si trovano in include e che sono minori di 1,5 k

```
find /usr/include/ -name 's*.h' -size -1536c 2>/dev/null | xargs -Ivar grep -oE '[[:alpha:]]+'
var | sort | uniq -c | sort -nr | head
```

NB se metti - davanti al numero di k escono quelli inferiori, se metti solo size ti stampa quelli con esattamente quei k

stampa parole con occorrenza più alta di file che hanno estensione .h che si trovano in include e che sono minori di 1,5 k e sono stati modificati meno di 2 settimane fa

```
find /usr/include/ -name '*.h' -size -1536c -mtime -14 2>/dev/null | xargs -Ivar
```

```
grep -oE '[[:alpha:]]+' var | sort | uniq -c | sort -nr | head
```

stampa file contenuti in include che iniziano per s e sono .h che contengono la parola "the"

```
find /usr/include/ -mtime -1000 -name 's*.h' 2>/dev/null | xargs grep the
```

calcola quante volte appaiono le parole DEFINE e INCLUDE nel totale dei file .c e .h

```
find / -name *.[ch] | xargs cat | grep -oe define -oe include -c
```

ps. se avessi voluto stampare quante volte compaiono define e quante volte compaiono include

```
find / -name *.[ch] | xargs cat | grep -oe define -oe include |sort | uniq -c
```

stampa il nome dei file.txt che contengono la parola Pipeline

```
find /home/aripizz/ -type f -name "*.txt" 2>/dev/null | xargs grep Pipeline -l 2>
/dev/null
```

Scrivere una pipeline che per ogni utente del sistema conti il numero dei file posseduti e scriva per ogni utente "Username N: numerofile", ordinandoli in modo decrescente rispetto al numero di file.

```
for user in $(awk -F':' ' print $1
' /etc/passwd); do fil=$(find / -user $user -type f 2> /dev/null | wc -l); echo "Username
$user:  $fil"; done | sort -k3nr
```

Scrivere una pipeline che calcoli quanti sono gli utenti con un numero di file > di 0

```
s=0; for user in $(awk -F':' ' print $1
' /etc/passwd); do fil=$(find / -user $user -type f 2> /dev/null | wc -l); if [ $fil
-gt 0 ]; then s=$(($s+1)); fi; done; echo $s
```

```
 for user in $(awk -F':' 'print $1
' /etc/passwd); do for file in $(find / -type f -user $user 2>/dev/null); do echo $user
$file $(cat $file | wc -w); done; done | sort -k3nr | head -n 1
```

Scrivere una pipeline per copiare files da una cartella all'altra mantenendo inalterati permessi

```
for f in $(ls cart1); do mv cart1/$f cart2; done
```

Scrivere una pipeline per trovare tutti i files eseguibili > 5k e evidenziare i 5 più grandi

```
  find / -type f -executable -size +5k 2> /dev/null | xargs du | sort -nr | head -n
5
```

Scrivere una pipeline per archiviare i files modificati nell'ultima settimana in cartellagang

```
  find / -mtime -7 -type f 2> /dev/null | xargs -Ivar cp var cartellagang
```

copiare tutti i file .h che contengono almeno 2 vote la parola "include" di /usr/include in una cartella

```
  mkdir carttt; for f in $(find /usr/include/ -type f -name '*.h'); do n=$(cat $f| grep
include | wc -l); if [ $n -gt 1 ]; then cp $f carttt; fi ; done
```

controllo nella cartella (pipe sopra) che ogni file contenga più di una volta la parola include e stampa il numero di volte in cui viene ripetuta

```
  for f in $(ls carttt/); do n=$(cat carttt/$f | grep include | wc -l); echo $n; done
```

Scrivere una pipeline che trovi i tre files più grossi per ogni utente

```
  for us in $(awk -F':' 'print $1' /etc/passwd); do echo $us; find -type f -user $us
2>/dev/null | xargs -0 du | sort -nr | head -3| awk 'print $2'; done
```

Scrivere una pipeline per archiviare tutti i files che contengono la parola "login"

```
  find / -type f 2>/dev/null | grep login |xargs tar -cvf logins.tar
```

Scrivere una pipeline che calcoli lo spazio occupato da ogni utente (dai suoi files)

```
  for user in $(awk -F':' 'print $1' /etc/passwd);do printf $user:  ; find / -user $user
-type f | xargs du -k | awk 's = s+ $1 END print s'; done 2>/dev/null
```

Trova i file modificati di venerdì

```
  find / -type f -printf "%Ta %p
n" | grep 'Fri.*'
```

Trova tutti i file che hanno righe dispari

```
  find / -type f 2>/dev/null | xargs wc -l | awk -F' ' 'if ( $1 % 2 != 0 ) print $2
" has" $1 " lines"'
```

Trova la somma del peso dei file per ogni utente

```
for user in $(awk -F':' 'print $1' /etc/passwd); do size=$(find / -user $user | xargs
du | awk 's=s+$1 END print s'); echo $user $size; done
```

Trovare tutti i file col numero dispari di righe (quindi sono i file di testo)

```
for f in $(find / -type f 2>/dev/null); do lines=$(cat $f | wc -l); if [ $(($lines
% 2)) -ne 0 ] ; then echo "$f :  $lines" ; fi ; done
```

Trovare tutti i file modificati di venerdi' (sia un venerdi' particolare che di venerdi' in generale)

```
find / -type f -printf "%p %Ta" 2>/dev/null | grep Fri | awk 'print $1'
```

Trovare tutti i link simbolici nel filesystem

```
find / -type l
```

Trovare tutti i file di testo che non siano script

```
find / -type f -not -executable | xargs ls -l
```

Quali file ha modificato l'utente XY in una specifica data?

```
find / -user user -printf "%p %TD" 2> /dev/null | grep 06/13/19 | cut -d ' ' -f1
find / -user user -printf "%p %TD" 2> /dev/null | grep 06/13/19 | awk 'print $1'
```

Quali file di testo ha modificato l'utente XY nell'ultimo anno?

```
find / -user user -type f -mtime -365 2>/dev/null
```

Trova i file non eseguibili

```
find ~/solab-jos/ -type f -not -perm /111 -printf "%M %p "
```

1)Conta il numero di include e define (maiuscoli e minuscoli) nei file .h o .c. nel sistema

grep il -i serve per ignorare la differenza tra maiscolo e minuscolo, - o stampa solo il match esatto che trova, se non lo mettessi mi metterebbe tutta la riga nella quale trovo il match

```
find / -type f -name "*.h" -o -name "*.c" 2>/dev/null| xargs cat| grep -i -o 'define\|include'| tr '[:upper:]' '[:lower:]' | sort | uniq -c
```

1.1) pipe che cerca i file txt e stampa il nome di file seguito dalle righe pari in essi contenute, awk in quetso caso lo uso per fare il controllo di parità delle righe

```
for file in $(find /home -type f -name "*.txt"); do echo $file ; cat $file | awk 'NR%2==0'; done
```

2)#Calcolare lo spazio occupato dai file con un numero di righe pari

Inizializzo s=0, scorro tutti file txt , stampo il nome del file seguito dal conteggio della linea (wc -l) e lo mette sulla stessa linea dato che uso printf. Finisce il for seleziona solo le righe che hanno il secondo elemento pari, e ne stampa il primo elemento (che è il nome del file), du metet accanto al nome la dimensione, selezionamo e stampiamo solo la dimensione e usiamo un' ultima volta awk per fare la somma sulla variabile s.

```
s=0; for file in $(find /home -type f -name "*.txt"); do printf $file" "; cat $file | wc -l  ; done | awk '$2%2==0' | awk '{print $1}'| xargs du | awk '{print $1}' |awk  '{s=s+$1} END {print s}'
```

3)#trovare tutti i file modificati di venerdì.

%p mi stampa il percorso del file trovato e ta la data dell'ultimo accesso. Con Grep vado a cercare (mi seleziona quelli con la parola Fri)

```
find / -type f -printf '%p %ta \n' 2>/dev/null | grep -E Fri | awk '{print $1}'
```

4)# Per ogni utente del sitema conti il numero di file posseduti e li ordini in maniera decrescente

awk lo uso per dividere sui due punti e stampo il primo parametro (non a video ma è il valore che faccio assumere alla variabile del for user), poi stampo la variabile user (il nome di ogni utente) , trovo tutti i file che hanno il propietario uguale allo user che sto considerando e li conto con wc -l (lo faccio per ogni user del ciclo), termino il for, e uso awk prima per dividere la stringa in input sullo spazio (il contenuto delle virgolette) e poi ristampo la striga con i parametri invertiti in modo tale da poterli ordinare con il sort (e nella terza riordino anche in modo da avere prima l'utenet e solo dopo il nuemro)

```
for user in $(find /etc/passwd | xargs cat |awk -F':' '{print $1}'); do printf $user" "; find / -type f -user $user 2>/dev/null | wc -l ; done | awk -F' ' '{print $2" " $1}'| sort -n
```

```
for user in $(find /etc/passwd | xargs awk -F':' '{print $1}'); do printf $user" "; find / -type f -user $user 2>/dev/null | wc -l ; done | awk -F' ' '{print $2" " $1}'| sort -n | awk -F' ' '{print $2" " $1}'
```

5) Trovare le parole nelle righe pari e queste parole devono avere il pattern vocale-consontante.vocale (precedente) (sono riuscito a farlo solo se il pattern è alla fine della parola)

cerco tutti i file, li ridirigo in standard output con cat, prendo le righe pari, prendo le parole con il primo grep (la o fa sì che io prenda solo le parole) , con l'altro grep richiedo la sequenza vocale consonante vocale.

```
for file in $( find /home/giona/Desktop/ -type f 2>/dev/null ); do cat $file |  awk 'NR%2==0' |  grep -ioE '[[:alpha:]]+' | grep -iE '(.*)([aeiou])([^aeiou])\2\1' ; done  2>/dev/null
```

6)Stampare i 3 file più grandi per ogni utente

```
for user in $(find /etc/passwd |xargs cat| awk -F ':' '{print $1}'); do echo $user; find / -type f -user $user 2>/dev/null |xargs  du 2>/dev/null| sort -n | tail -3 ; done
```

7)Copiare i file di giona e che iniziano per vocale da una cartella all'altra, mantenendo inalterati i permessi

Versione 1 che per controlalre che inizi con vocale il nome del file impone che prima del nome del file ci debba essere tutto il percorso

```
mkdir landing; find /home/giona/Desktop/lab/ -type f -user giona 2>/dev/null| grep -iE '(/home/giona/Desktop/lab/)([aeiou])(.*)'| xargs -I var cp --preserve var /home/giona/Desktop/landing/
```

Con questa invece riesco ad eliminare il dover mettere il percorso, ma poi non funziona la copia dato che ho rimodellato i file e non sono più un percorso completo...

find /home/giona/Desktop/lab/ -type f -user giona 2>/dev/null|awk -F '/' '{print $6}' | grep -iE '(^[aeiou])(.*)'

8)Pipe che usa sed, molto utile per sostituire delle parole, in questo caso sostituisce copia (seguito da uno spazio con nulla), in sed la -e (expression) e la g

find /home/francesco/Scrivania/EserciziSisop -user francesco -type f -name "*.txt" 2>/dev/null | awk -F'/' '{print $6" "$7}' | sed 's/copia //'

9)Stampare la dimensione ed i file con dimensione compresta tra 8MB e 2GB

il bh alla fine fa sì che sua human-reable

find /home/giona/Desktop/lab -type f -size +8M -size -2G |xargs du -bh

10)Comprimere tutti i file di dimensione compresa tra 2MB e 2GB

c=create z=formato zip

find /home/giona/Desktop/lab -type f -size +2M -size -2G |xargs tar -cvzf compresso.tar

11)Pipeline che legge il file testo.txt nel percordo nel quale ci troviamo e legge le parole, le ordina, le conteggia, le riordina per apparizione in maniera decrescente

pwd | xargs -I var find var/testo.txt | xargs -I var cat var | grep -ioE '[[:alpha:]]+' | sort | uniq -c | sort -nr

12)Pipeline che cercano i file che contengono la parola casa e stampa il nome dei file che soffisfano la condizione

find /Users/giona/Desktop -type f -name '*.txt' |xargs grep -iE 'casa'

oppure con il for (ma non stampa il contenuto del file)

for file in $(find /home/francesco/Scrivania/EserciziSisop -type f -printf "%p \n"); do echo $file | xargs grep -ie 'ciao' -l; done

13)Trova tutti i file png e li cambia in jpg, fino a prima di bash creo delle strinhe che sono come un comando che fa move del primo elemento nel secondo

find /home/giona/ -name '*.png' | sed -e 's/.*/mv & &/' -e 's/png$/jpg/'| bash

FIND: find ... -print0 | xargs -0 ...

cp -p --parents {} ./localinclude

(-p -> preservare permessi | --parents -> preserva struttura directory)

ps -eo uname,pid,ppid,args

- Creazione di un archivio€(€create)

tar -cvf <archive_name>.tar <files>

- Esaminare il contenuto di un tarball (file *.tar). Stampa lista del contenuto :

tar -tvf <archive_name>.tar

- Estrarre I file dall'archivio (extract)

tar -xvf <archive_name>.

- Uso congiunto find e tar (mediante find ... -exec ...). Il parametro r di tar aggiunge (uno alla volta) I file trovati da find all'archivio senza ricrearlo.

find /usr/include/ -name 's*.h' -exec tar -rvf file.tar {} \;

- Creazione di un archivio (create) e compressione con gzip (z)€:

tar -cvzf <archive_name>.tar.gz <files>

- Estrarre I file dall'archivio (extract) e decompressione gunzip (z) :

tar -xvzf <archive_name>.tar.gz

(per testare usare t al posto di x)

- Creare un archivio tar.gz contenente tutti i file la cui dimensione è minore di 50 KB

find / -type f -size -50k | tar --exclude "/dev/*" --exclude "/sys/*"€--exclude "/proc/*"

-cz -f test.tgz -T -tar -ztvf test.tgz

- Calcolare spazio occupato da ogni utente

for target in $(awk -F':' '{ print $1}' /etc/passwd);

do res=$(find / -user $target -type f -printf '%p|%s\n' 2>/dev/null |€

awk -F\| '{ s += $2 } END {if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}');

[[$res != ""]] && echo "$target $res"; done | sort -k3 -n -r | head -n 3

- Trovare i file con nome dispari e stampare il nome del file e il numero di caratteri (viene un carattere in più perchè conta lo \n)

for file in $(find . -name '*.txt'); do n=$(echo $file | awk -F/ '{print $NF}' | awk -F'.' '{print $1}' | wc -m); if [[$n%2 -ne 0]]; then echo $file " " $n; fi; done

- Trovare i file...Versione corretta

for file in $(find . -name '*.txt' | awk -F/ '{print $NF}' | awk -F'.' '{print $1}');

do n=$(echo ${#file}); if [[$n%2 -ne 0]]; then echo $file " " $n;fi; done

- Trovare i file .c o .h con nome dispari di dimensione minore di 3k e stampare quanti sono

for file in $(find / -type f -name '*.[ch]' -size -3k 2> /dev/null | awk -F/ '{print $NF}' | awk -F'.' '{print $1}'); do n=$(echo ${#file}); if [[$n%2 -ne 0]]; then echo "$file"; fi; done 2> /dev/null | wc -l

- Per ogni utente stampare la somma di dimensione dei file (fatto con xargs du)

for user in $(awk -F':' '{print $1}' /etc/passwd); do size=$(find / -user $user -type f 2> /dev/null | xargs du -bc | tail -1 | awk '{print $1}'); echo "$user $size"; done

ESERCIZI PIPELINE

1-Scrivere una pipeline di comandi che identifichi le informazioni sul processo dropbear

```
ps aux | grep  ‘dropbear’
```

2-Scrivere una pipeline che identifichi il processo con ppid più alto

```
ps -eo uname,pid,ppid,args | sort -nk3 | tail -n1
```

3-Trovare il numero totale di file contenuti in /usr/bin e /var

```
expr $(find /usr/bin -type f | wc -l) + $(find /var -type f | wc -l)
```

4-Supponendo di avere il codice di un programma in un linguaggio in cui i commenti occupino intere righe e vengano indicati con #, allora scrivere una pipe che mostri il programma senza commenti.

```
cat nomefile | grep -v ‘#’
```

5-trovare la memoria occupata in byte da /usr/bin e /var insieme

```
expr $(du -sb /usr/bin | cut -f1) + $(du -sb /var | cut -f1
```

6-Trovare tutti i file presenti in /usr/include/ che hanno estensione .h. Prima di ogni riga di output rendere lo spazio un solo togliendo l’ultima riga

```
find /usr/include/ -type f -name '*.h' | xargs wc -l | sed 's/ \
+/ /' | grep -v 'total'
```

7-Trovare il file più grande in un determinato path

```
find path -type f | xargs du | sort -rn -k1 | head -n1
```

8-Copiare alcuni file (ad es. il cui nome segue un certo pattern) di un ramo(in questo caso /usr/include/ in un altro(creato manualmente da noi es: /localinclude) mantenendo la gerarchia delle directory

```
find /usr/include/ -name 's*.h' | xargs -I{} cp --parents {}
./localinclude
```

9-Calcolare lo spazio occupato dai file di proprietà di un certo utente

```
find /home -user user -type f | xargs du -k | awk '{ s = s+$1 }
END {print " Total used: ",s}'
```

10-Cambiare estensione ad ogni file .png in .jpg.

```
find ./ -type f -name '*.png' | sed -e 's/.*/mv & &/' -e
's/png$/jpg/' | bash
```

11-Creare un archivio tar.gz contenente tutti i file la cui dimensione è minore di 50 KB
sudo tar -cvf prova.tar $(find . -type f -size -50k)

12-Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano più spazio nel sistema.
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find / -user $target -type f -printf '%p|%s\n' 2>/dev/null | awk -F\| '{ s += $2 } END {print "Sum:",s}'); [[$res != ""]] && echo "$target $res"; done | sort -k3 -n -r | head -n 3

13-Calcolare il numero di directory di profondità 1 a partire da / e /opt
wc -l <(find / -mindepth 1 -maxdepth 1 -type d) <(find /opt -mindepth 1 -maxdepth 1 -type d)

14-Stampa i nomi di tutti gli utenti presenti nel sistema
awk -F':' '{ print $1}' /etc/passwd

15-Estrarre tutti gli indirizzi IP nei file contenuti nel ramo etc del filesystem:
Variante 1: find /etc/ -type f -exec grep -Eo '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' {} \;
Variante 2: for file in $(find /etc/ -type f); do cat $file | grep -Eo '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'; done
Variante 3: for i in $(find -type f); do grep -vi '[a-z]' $i | grep -E '([0-9]{1'3}\.){3}[0-9]{1,3}'; done;

16-Trovare tutti i file presenti in /usr/include/ che hanno estensione .h contando le linee e facendo sì che prima di ogni riga di output ci fosse un solo carattere di spazio, togliendo infine l'ultima riga relativa al totale
find /usr/include/ -type f -name '*.h' | xargs wc -l | sed 's/ \ +/ /g' | grep -v 'total'

17-Trovare tutti i file presenti in /usr/include/ che hanno estensione .h il cui inode è pari e il numero di righe dispari:
for target in $(find /usr/include -type f -name '*.h'); do inode=$(ls -i $target | awk '{print $1}'); ln=$(wc -l $target | awk '{print $1}'); if ["$(expr $inode % 2)" -eq "0"] && ["$(expr $ln % 2)" -eq "1"]; then echo "$target inode: $inode numero righe: $ln"; fi; done;

18-Trovare tutti i file con i numeri dispari di righe

for f in $(find -type f -name "s*"); do nr=$(wc -l $f | awk '{print $1}'); if [$(expr $nr % 2) -eq 1]; then echo $f; else echo "NO";fi;done

19-Quanta memoria occupa il processo più grande
pm=$(ps aux | sort -nk4 | tail -n1 | awk '{print $4}'); echo $pm

20-Trovare tutti i file di testo che non siano script
for f in $(find -type f); do strtc='#!/bin/sh'; if [$(cat $f | head -n1 | awk '{print $1}') != $strtc]; then echo $f; fi; done;

Ottenere l'inode di un file: ls -i nomefile

strace -e trace=memory ls -> vengono tracciate le system call che hanno a che fare
con la gestione della memoria

strace -e trace=memory ls 2>&1 | grep ‘(‘ -> filtriamo le righe con (per trovare le syscall

21- Trovare tutti i file modificati di venerdi' (sia un venerdi' particolare che di venerdi' in generale)

```
for x in $(find . -type f)
do
        if [[ $(date -r $x | awk '{print $1}') == 'ven' ]]; then
                echo $x
        fi
done
```

22-Calcolare quante volte appaiono le parole define e include nel totale dei file .c e .h (Intendo che devo ottenere il conteggio finale per la parola define e per la parole include)
sudo find / -type f -name '*.c' -o -name '*.h' | xargs -I{} grep -Eo 'define|include' {} | sort | uniq -c

23-Qual è la parola più frequente in string.h?
grep -o '[A-Za-z]*' /usr/include/string.h | tr A-Z a-z | sort | uniq -c | sort -r

24-Trovare l'utente col file piu' recente nel sistema (escludo Root. Ordina già ls con -t)
ls / -lRta 2>/dev/null| grep '^-' | grep -v 'root' | head -n1

1 - Trovare il file più grosso di un certo ramo

```
find [ramo] -type f | xargs du | sort -rn | head -n1
```

2 - Copiare i file con caratteristiche x del ramo origine nel ramo destinazione mantenendo la gerarchia delle directory

```
find /origine -type f -name “x” -print0 | xargs -0 -I{} cp --parents {}
./destinazione
```

3 - Calcolare lo spazio occupato dai file di proprieta’ dell’utente root e mostrarlo in modo puccioso

```
find / -user root -type f -exec du -sk {} \; | awk '{s=s+$1} END {print
"Total used: ",s }'
```

4 - Scrivere un comando che conta quanti file ci sono in un determinato ramo del filesystem

```
find /home -type f | wc -l
```

5 - Creare un archivio tar.Gz contenente tutti I file la cui dimensione e’ minore di 50 kb

```
tar -cvf prova.tar $(find . -type f -size -50k)
```

6 - Rinominare un certo numero di file: da png a jpg

```
find . -type f -name '*.jpg' | sed -e 's/.*/mv & &/' -e 's/jpg$/png/' | bash
```

7 - Creare un file da 10MB costituito da caratteri casuali (/dev/urandom) e verificare se contiene la parola jos

```
dd if=/dev/urandom of=fileOut bs=1k count=10240 ; grep --binary-files=text
-Eo '.{0,5}JOS.{0,5}' fileOut
```

8 - Trovare l’utente che ha il maggior numero di file nel sistema

```
for user in $(cut -d: -f1 /etc/passwd); do find / -type f -user $user
2>/dev/null | echo -e $user:'\t' $(wc -l); done | sort -rn -k 2 | head -n 1
```

9 - Trovare I 3 utenti che, sommando la dimensione dei loro file, occupano piu’ spazio nel sistema

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find / -user
$target -type f -printf '%p|%s\n' 2>/dev/null | awk -F \| '{ s += $2 } END
{if(s!=0){print "Sum:",s}else{print "Sum: 0"}}'); [[ $res != "" ]] && echo
"$target $res"; done | sort -k3 -n -r | head -n 3
```

10 - Tutti i file in un ramo, inode pari e numero di righe dispari

```
ls -laih | awk 'NR % 2 != 0' | grep -E "^[0-9]*[0,2,4,6,8]\b"
```

11 - Trovare i nomi di ogni syscall per un comando

```
strace ls 2>&1 | grep '(' | sed "s/(.*//" | sort -u

strace -e trace=memory ls 2>&1 | grep '(' | sed "s/(.*//" | sort -u
```

12 - Elencare tutti gli indirizzi ip di un ramo

```
grep -roEh
'\b((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(\.|$)){3}(25[0-5]|2[0-4][0-9]|[01]
?[0-9][0-9]?)\b' /home/triad/Desktop 2>/dev/null | sort -u

grep -roEh '[0,9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' /etc 2>/dev/null
| sort -u
```

Commands qemu-gdb

qemu-system-i386 -S -gdb tcp::42000
target remote localhost:42000

nasm -f bin -o [nomeProgrammaOut.bin] [nomeProgrammaIn.asm]
qemu-system-i386 -S -hda [nomeProgrammaOut.bin] -gdb tcp::42000
x/x 0x7c00
b* 0x7c00
c
set architecture i8086
x/lli 0x7c00