Lab. di Sistemi Operativi
- Esercitazione n° 2-

- comandi concatenati -
- comando di ricerca: grep -

Sommario: comandi concatenabili

Solo a **inizio** pipe: `echo`, `ls`, etc.
(tutti quelli che scrivono su stdout)

Anche al **centro**: `wc`, `sort`, `uniq`, `grep`, `cat`, `head`, `tail`

- se richiamati senza argomenti, leggono da stdin
- scrivono su stdout

Solo a **fine** pipe: `less` (paginatore interattivo)

Sort

sort [*options*] [*file...*]

permette di (ri)ordinare o fondere insieme il contenuto dei file passati come parametri, oppure di (ri)ordinare le linee passategli in input.

In assenza di opzioni che definiscano diversi criteri di ordinamento, quest'ultimo avviene in base al primo campo ed è alfabetico.

Alcune opzioni:

-f	ignora le differenze tra lettere minuscole e maiuscole
-n	considera numerica anzichè testuale la chiave di ordinamento
-r	ordina in senso decrescente anzichè crescente
-o *fileout*	invia l'output a fileout anzichè sull'output standard
-t *s*	usa s come separatore di campo
-k *s1,s2*	usa i campi da s1 a s2-1 come chiavi di ordinamento

Esercizio n° 1

- Realizzare una combinazione di comandi unix per visualizzare l'ultimo file in ordine alfabetico presente nella directory di lavoro

Soluzione

ls | sort –r | head -1

Esercizio n° 2

- Ordinare le righe del file di testo **votoStudenti.txt** (anche in senso inverso):

```
Gianni      20
Bruno       15
Carlo       10
Alice       30
```

Soluzione: ordinamento alfabetico default

```
sort votoStudenti.txt
```

```
Alice 30
Bruno       15
Carlo       10
Gianni      20
```

Soluzione Esercizio n° 4

Soluzione: ordinamento inverso

```
sort -r votoStudenti.txt
```

```
Gianni      20
Carlo 10
Bruno       15
Alice 30
```

inverte il senso di ordinamento

Soluzione: ordinamento inverso su file

```
sort -r votoStudenti.txt -o file_risultato.txt
```

ordinamento inverso con scrittura del risultato nel file _risultato.txt invece che sullo standard output

Esercizio n° 3

- Realizzare una combinazione di comandi unix che, dato un file di testo esistente **elenco.txt**, crea un nuovo file **nominativo.txt**, contenente la riga di **elenco.txt** che viene lessicograficamente per seconda.

Ad esempio se il contenuto di un file **elenco.txt** è il seguente:

valeria
aldo
roberta
bruno
sandro
paola

il comando deve creare il file **nominativo.txt** con il seguente contenuto: **bruno**

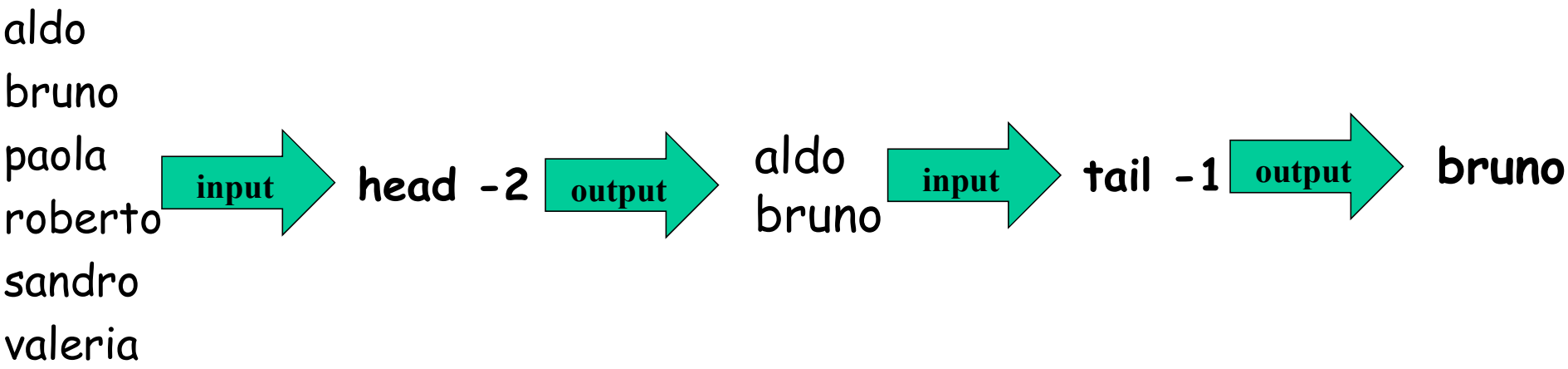
Soluzione Esercizio n° 3

Soluzione:

```
sort elenco.txt | head -2 | tail -1 > nominativo.txt
```

ordina il contenuto del file **elenco** in ordine alfabetico

scrive il risultato sul file **nominativo**



- comando sort: con chiavi di ordinamento -

- sort: chiavi di ordinamento -

- Tramite l'opzione -k è possibile definire una chiave di ordinamento su una porzione della linea contenuta nel file

- Specificando più opzioni -k è possibile definire più chiavi di ordinamento da usare in cascata.

Sintassi sort con opzione -k

sort –k campo_inizio [tipo] [,campo_fine [tipo]]

campo_inizio, campo_fine

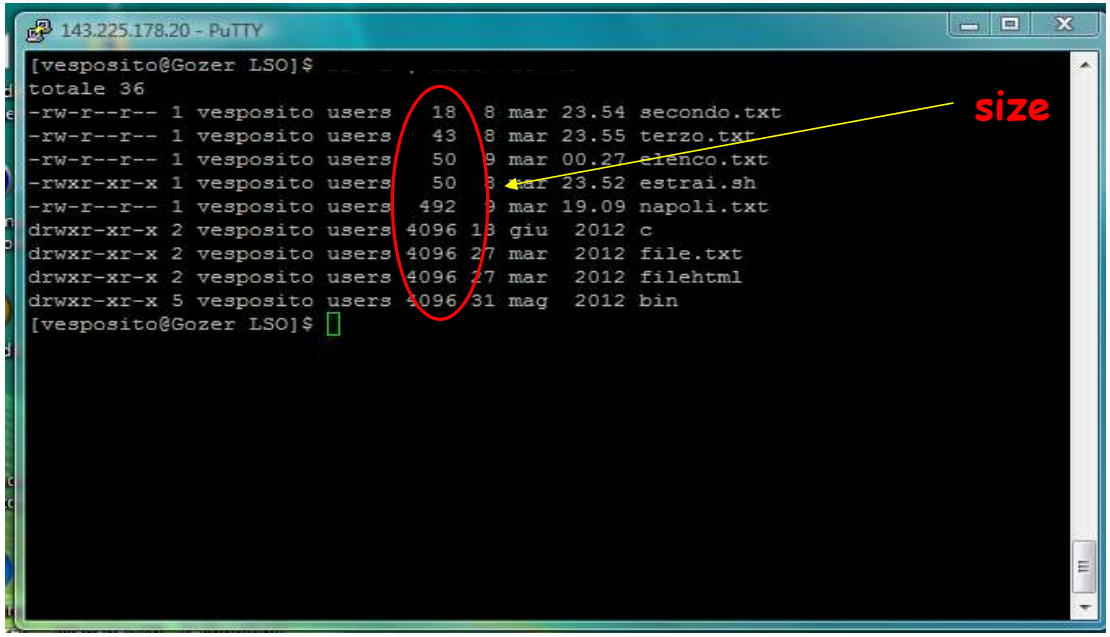
chiavi per restringere l'ordinamento su una porzione di linea

tipo

è il tipo di ordinamento applicato sui campi specificati

Esercizio n° 4

- Realizzare una combinazione di comandi Unix per visualizzare i file presenti nella directory di lavoro con un ordine di dimensione crescente.



Soluzione: size crescente

ls -l | sort -k5 -n

Esercizio n° 5

- Applicare al file **votoStudenti.txt** un ordinamento numerico con **chiave**:

Gianni 20 ← **Secondo campo numerico**

Bruno 15

Carlo 10

Alice 30

chiave di ordinamento: secondo campo del file da ordinare

```
sort -k2 -n esempio.txt
```

output

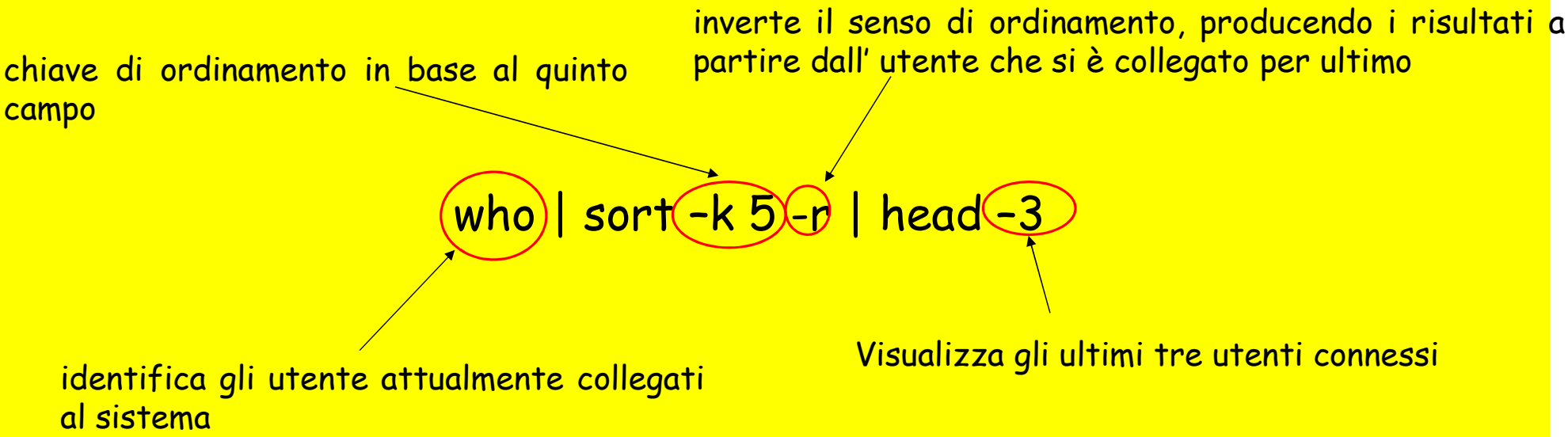
```
Carlo 10
Bruno 15
Gianni
20
Alice 30
```

Ordina numericamente in quanto il campo scelto come chiave di ordinamento è numerico.

Esercizio n° 6

- Realizzate una combinazione di comandi Unix per visualizzare gli ultimi **n** (numero scelto) utenti collegati.

Soluzione



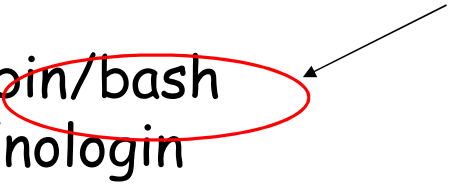
Ricordiamo che…

- Le righe di **/etc/passwd** si presentano nella seguente forma:

Username:Password:UserID:GroupID:Info:HomeDirectory:Shell

- Esempio:

Indica la shell di default per quell' utente.



root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
[…]
diego:x:501:503::/home/diego:/bin/bash

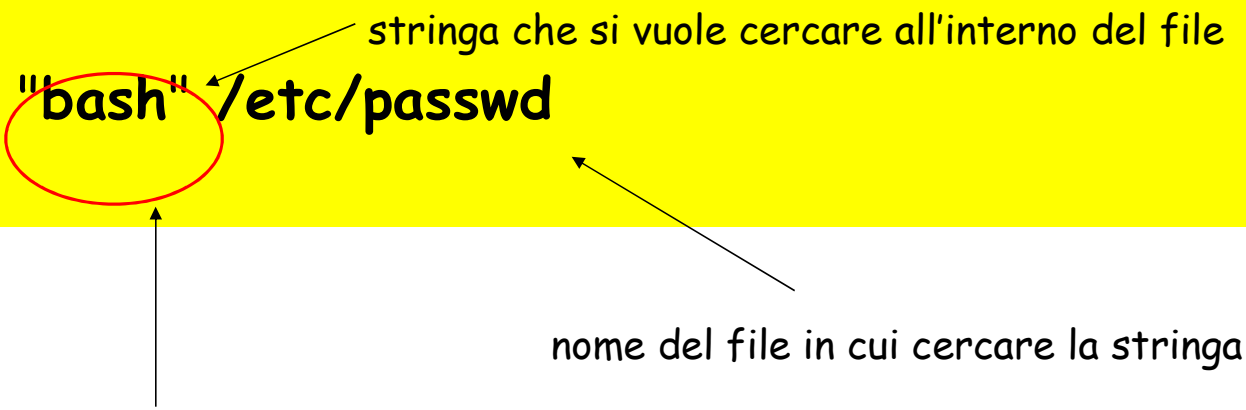
Esercizio n° 7

- Scrivere un comando Unix che visualizza tutti gli utenti diversi contenuti nel file **/etc/passwd**, che usano **bash** come shell di default

Sintassi

grep [opzioni] "stringa" nome_file ...

Soluzione



visualizza tutti gli utenti in /etc/passwd che hanno come shell di default bash.

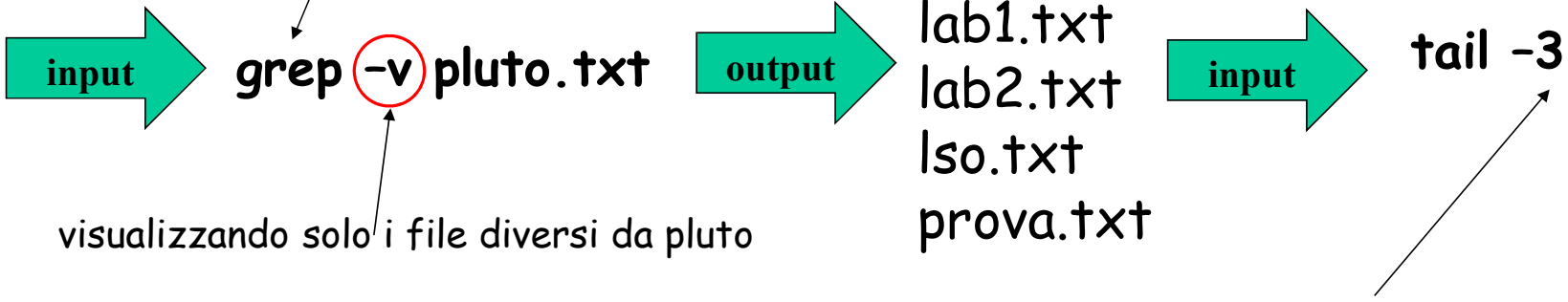
Esercizio n° 8

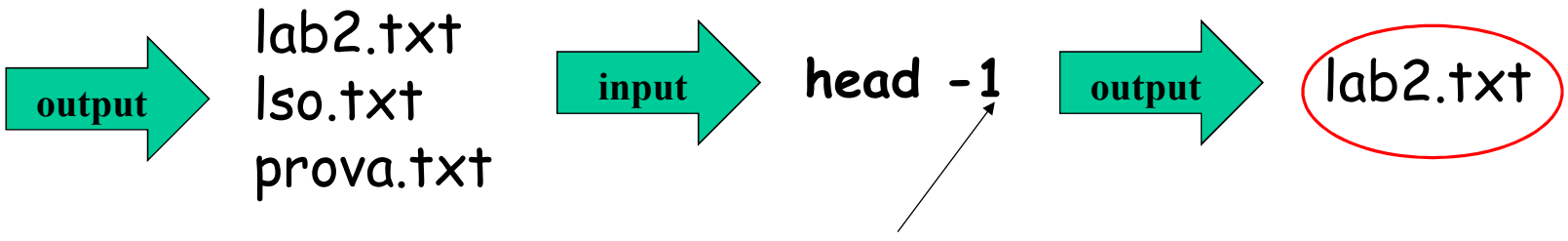
- Realizzate una combinazione di comandi Unix per visualizzare il terzultimo file presente nel sistema diverso dal file di nome **pluto.txt**, secondo il normale ordinamento prodotto dal comando che permette di visualizzare tutti i file presenti nel sistema.

Soluzione Esercizio n° 8

Primo comando: **ls -l** ottengo la lista dei file nella directory di lavoro

lab.txt
lab1.txt
lab2.txt
lso.txt
pluto.txt
prova.txt





Soluzione Esercizio n° 8

Soluzione

Elimina dalla lista dei file passati in input il file di nome pluto.txt

```
ls -l | grep -v pluto.txt | tail -3 | head -1
```

mostra tutti i file

pipe

Visualizza le ultime tre righe

Visualizza la prima riga

Esercizio n° 9

- Nella propria work directory e in tutte le sottodirectory elencare utilizzando un comando Unix tutti i file con estensione ".txt" (file di testo).

cerca nella directory corrente e
nelle sottodirectory i file che terminano con .txt

Soluzione con (ls + grep)

```
ls -R | grep "\.txt$"
```

cerca nella directory corrente che ha il nome speciale "**punto**"

Soluzione con find

```
find . -name "*.txt"
```

al –name usando il carattere jolly "*" si passa non il singolo nome del file ma tutti i file con estensione "txt"

elenca nel path (directory corrente) tutti i file con estensione .txt

Esercizio n° 10

- Nella propria work directory creare un file out.txt che contiene tutti i nomi dei file regolari con permesso di eseguibilità per l'utente.

Soluzione con (ls + grep)

```
ls -l | grep -?x.*
```

Esercizio n° 11

- Dato il file out.txt creare il file result.txt che contiene tutte le righe di out.txt che finiscono con 'a' e che contegono una 'b'.

Soluzione con (cat + grep)

```
cat out.txt | grep b.*a$ > result.txt
```

Esercizio n° 12

- Dato un file test.txt, creare un file output.txt che contiene tutte le righe di test.txt escludendo tutte le righe di test.txt che contengono almeno un punto o una virgola.

Soluzione con (cat + grep)

```
cat test.txt | grep -v .*\..* | grep –v .*,.* > output.txt
```

Esercizi

1. Elencare i file della directory corrente in ordine alfabetico *inverso*
2. Contare i file della directory corrente che contengono una “z” nel nome
3. Scrivere nel file “elenco” l'elenco dei file nella directory corrente, in ordine alfabetico
4. Creare un file che si chiami come l'utente corrente
5. Creare un file che si chiami come l'host corrente, e che contenga il nome dell'host corrente

Sommario: le sostituzioni

- Variabili: `${variabile}` (*parameter expansion*)
- Caratteri jolly: `* ?` (*filename expansion*)
- *Command substitution*: `$(comando)`

Vengono eseguite in quest'ordine

Esercizio: come si verifica in che ordine vengono eseguite?

Esercizi

1. Assegnare alla variabile x il numero di righe di un file a vostra scelta
2. Assegnare alla variabile x l'elenco dei file che cominciano con un punto
3. Per ogni parola contenuta nel file pippo.txt, creare un file con nome uguale a quella parola

Word splitting

Dopo aver effettuato parameter expansion o command substitution, la shell effettua la **suddivisione in parole**

La variabile IFS (internal field separator) definisce i separatori

(di default, IFS="<space><tab><newline>")

Come effetto collaterale, il word splitting sostituisce i newline con spazi

In una directory con molti file, confrontare l'output di "ls" con quello di "echo $(ls)"

Word splitting

Consideriamo il programma C **myecho**:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  for (int i=0; i<argc ;i++)
    printf("argv[%d] = \" %s \" \n", i, argv[i]);

  return 0;
}
```

> myecho prova questo
argv[0] = “myecho”
argv[1] = “prova”
argv[2] = “questo”

Word splitting

> myecho prova questo
argv[0] = “myecho”
argv[1] = “prova”
argv[2] = “questo”

> myecho “prova questo”
argv[0] = “myecho”
argv[1] = “prova questo”

> myecho $(echo “prova questo”)
argv[0] = “myecho”
argv[1] = “prova”
argv[2] = “questo”

Cambiare il separatore di parole:

> temp=$IFS
> IFS=”v”
> myecho $(echo “prova questo”)
argv[0] = “myecho”
argv[1] = “pro”
argv[2] = “a questo”

> myecho “prova questo”
argv[0] = “myecho”
argv[1] = “prova questo”

> IFS=$temp