# Bash

## Sistemi Operativi e Laboratorio

A. Formisano

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Anno Accademico 2023-24

## Contenuti

# Interfaccia con l'Utente

**Interprete di comandi**:

A volte integrato nel kernel, più spesso è un programma/processo vero e proprio, detto shell.
Acquisisce un comando in forma di stringa di caratteri, lo decodifica e lo esegue. Ciò può causare l'esecuzione di un frammento di codice della shell stessa (comando built-in) o l'invocazione di un programma (solitamente di sistema, tipico nelle shell di UNIX/Linux)

Varie possibilità: sh, csh, ksh, **bash**, ...

Interfaccia grafica:

L'interfaccia è visuale e modella una scrivania virtuale in cui programmi, file, directory, ecc. sono rappresentati da oggetti grafici. Combina e coordina l'uso di diversi dispositivi, quali tastiera, schermo, mouse, ecc.





# Autenticazione, Utenti e Gruppi

- Unix/Linux è un sistema multi-utente: serve un meccanismo di autenticazione
- ad ogni utente è associato un username (e una password) e un *User IDentifier* UID
- gli utenti possono far parte di uno o più group (identificati da un *Group IDentifier* GID)
- ogni utente (e gruppo) gode di opportuni privilegi, root è l'utente che ha massimi privilegi
- ad ogni utente (solitamente) viene assegnata una home directory all'interno del file system
- dopo l'autenticazione l'utente interagisce con il sistema tramite l'interfaccia utente (ad es., la bash o una GUI)
- ogni processo in esecuzione (compresa l'interfaccia utente) ha associata la propria present working directory (directory corrente) pwd (quindi, ogni utente "lavora in una specifica directory" del file system (la pwd))

# Il file system, in breve

- È la parte del S.O. che implementa le funzionalità per la memorizzazione dei dati
- organizzato in:
  - un insieme di file, le unità di base di memorizzazione dei dati
  - directory, un meccanismo di organizzazione dei file
  - altri elementi con funzioni speciali, ad esempio associati a device, dispositivi fisici, risorse SW del SO
- intuitivamente diciamo che la directory "contiene" dei file (e delle altre directory)
- tipicamente, il F.S. è organizzato secondo una struttura "ad albero"
- la radice di tale albero è la directory root (radice), denotata da /





# Il file system, in breve

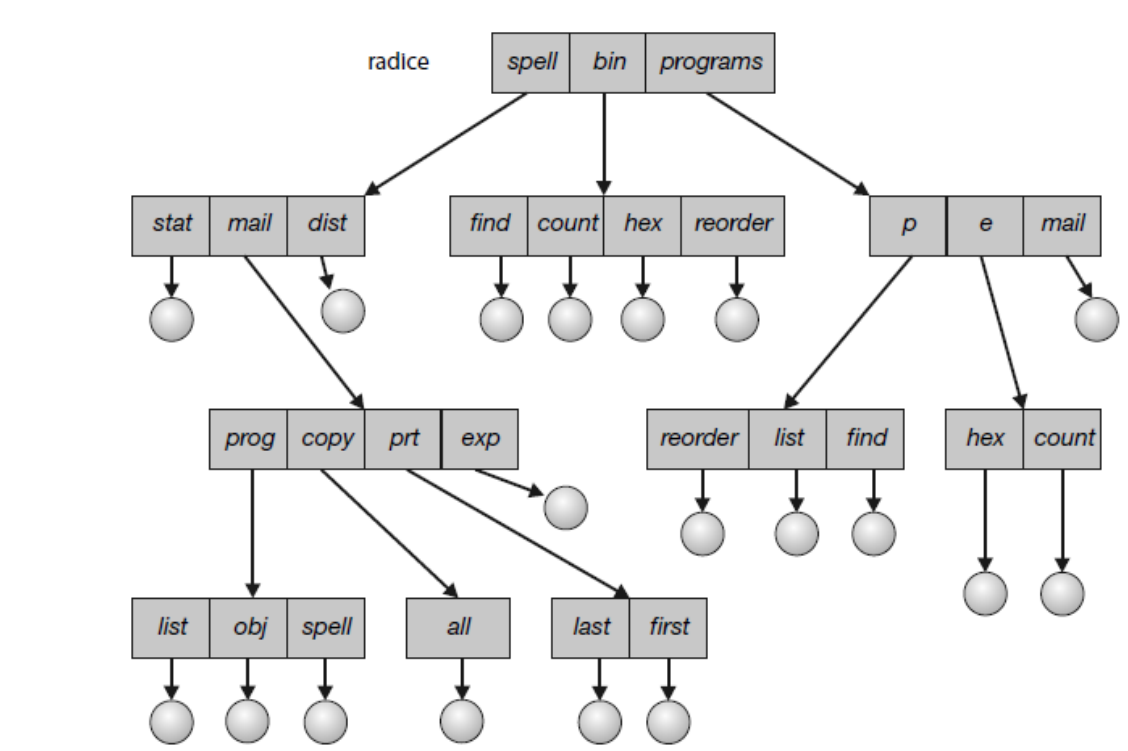



**Figura 13.9** Struttura della directory ad albero.

# Il file system, in breve

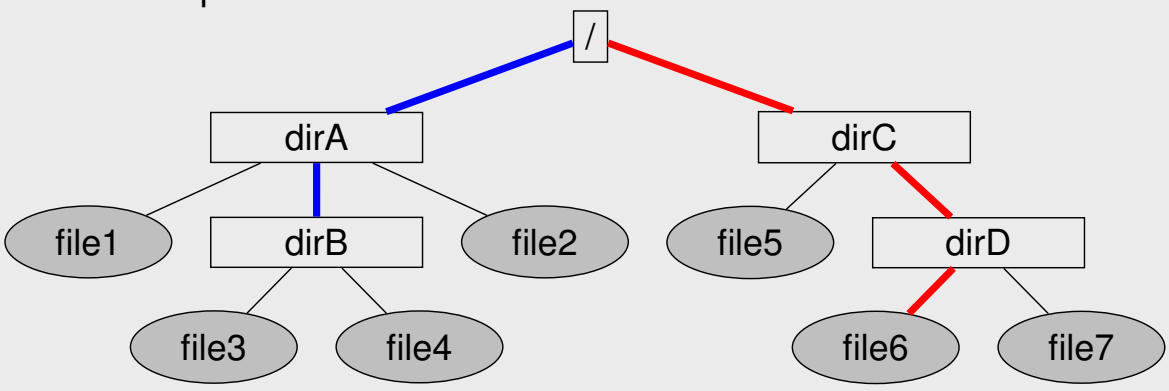
- ogni elemento nel file system viene univocamente individuato tramite *il percorso che dalla root conduce all'elemento stesso*
- tale percorso è detto pathname assoluto
- ogni pathname assoluto è composto dalla successione delle directory (i nodi interni dell'albero) che si deve *attraversare* per giungere all'elemento
- quindi tutti i pathname assoluti iniziano con /, a indicare la directory root





# Absolute pathname

Alcuni esempi:

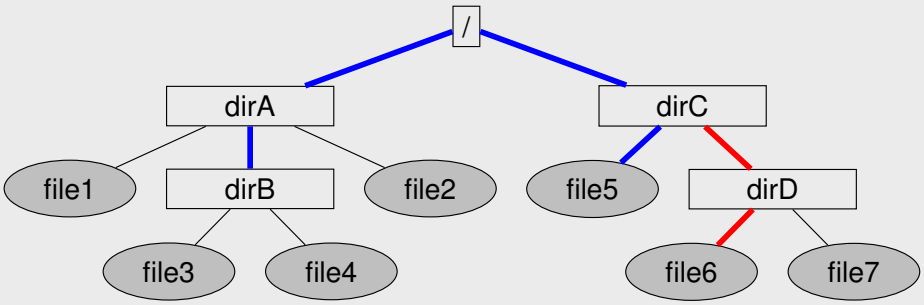



/dirA/dirB/
(o /dirA/dirB )

/dirC/dirD/file6

# Relative pathname

- Ogni posizione nel file system (che sia un file o una directory) può essere individuata tramite percorsi relativi ad altre directory
- tali percorsi sono detti pathname relativi
- ad esempio: il file6 è (anche) individuato dal pathname dirD/file6 *relativamente* alla directory dirC
- speciali pathname relativi sono . e ..
- file5 è (anche) individuato da ../../dirC/file5 a partire da dirB, o anche da ./file5 a partire da dirC








# Link

Tra gli elementi "speciali" presenti in un file system ci sono i link.
In unix/linux si distingue tra

- symbolic link: dei "collegamenti" ad altri punti del filesystem.
  In pratica sono dei file di testo contenenti il pathname dell'elemento "puntato" (target)
- hard link: sono dei "nomi alternativi" per lo stesso file memorizzato su disco
- le operazioni (ad es. l'apertura in lettura o in scrittura) effettuate "sul link" sono solitamente eseguite sull'elemento target
- la cancellazione di un symbolic link non elimina il target ma solo il link.
  (Ciò non è sempre vero per la cancellazione di un hard link)

(vedremo nelle prossime lezioni maggiori dettagli sugli hard link...)

# Comandi della shell

Possono essere di due tipologie

comandi builtin:

vengono eseguiti dalla shell stessa tipicamente corrispondono a funzionalità implementate dal codice della shell
La shell inoltre fornisce un linguaggio di scripting, i cui costrutti sono interpretati dalla shell stessa

comandi esterni:

sono comandi che corrispondono a file eseguibili memorizzati nel file system

- eseguiti da processi diversi dalla shell (solitamente processi figli della shell)
- possono riguardare funzionalità del SO, utilità, applicativi o programmi utente





# Interazione con la shell

L'interfaccia fornita dalla shell è a linea di comando, ovvero:

- la shell presenta un prompt (ad es. `user$`) ed attende la digitazione di un comando
- analizza il comando per determinare la correttezza della sintassi ed individuare eventuali argomenti
- se il comando è builtin esegue la procedura corrispondente
- se non viene riconosciuto come builtin e se il comando è specificato da un pathname, si individua tale file, oppure
- si cerca nel file system un eseguibile il cui nome corrisponda al comando invocato
- la ricerca avviene limitatamente alle directory indicate in una particolare variabile: PATH
- in caso positivo si esegue il file eseguibile, altrimenti si comunica un errore
- la shell ripropone quindi il prompt

# Parametri dei comandi

Consentono di specificare

- file o directory su cui il comando deve agire. Si specificano attraverso pathname assoluti o relativi (il semplice nome del file viene interpretato come pathname relativo alla pwd)
- dati o valori necessari per eseguire il comando. Sono stringhe di caratteri
- switch o opzioni per specializzare l'operato del comando e specificarne le modalità di esecuzione.
  Usualmente hanno due possibili sintassi:
  - **forma short:** iniziano con il simbolo **-** seguito da un carattere (ad esempio `-a`, `-h`). Più opzioni di questa forma possono essere (solitamente) compattate, come in `-akr`, che corrisponde a scrivere `-a -k -r`
  - **forma long:** iniziano con **--** seguito da una stringa (ad esempio `--help`, `--out`)





# Esempi di comandi

I seguenti sono alcuni comandi informativi

- **man**
  Sintassi: **`man`** `stringa`
  Visualizza la pagina del manuale relativa alla `stringa` indicata come parametro (solitamente `stringa` è a sua volta il nome di un comando)
- **whatis**
  Sintassi: **`whatis`** `stringa`
  Visualizza una descrizione sintetica (una riga) relativa al comando `stringa` (la prima riga della pagina del manuale)
- **apropos**
  Sintassi: **`apropos`** `stringa`
  Visualizza la descrizione sintetica (una riga) di tutti i comandi che contengono `stringa` nella loro descrizione sintetica

# Il comando `ls`

**sintassi:** **`ls`** `[opzioni] [pathname]`
Fornisce informazioni sul contenuto di directory

- `pathname` è un elenco di pathname separati da spazi, se omesso si usa la pwd
- alcune delle principali `opzioni` sono:
  - `-a` non ignora gli elementi il cui nome inizia con `.`
  - `-l` elenca informazioni dettagliate
  - `-R` processa ricorsivamente anche le sotto-directory
  - `--color` usa colori diversi a seconda del tipo di file
  - `--help` produce una descrizione sommaria del comando
- esempi:

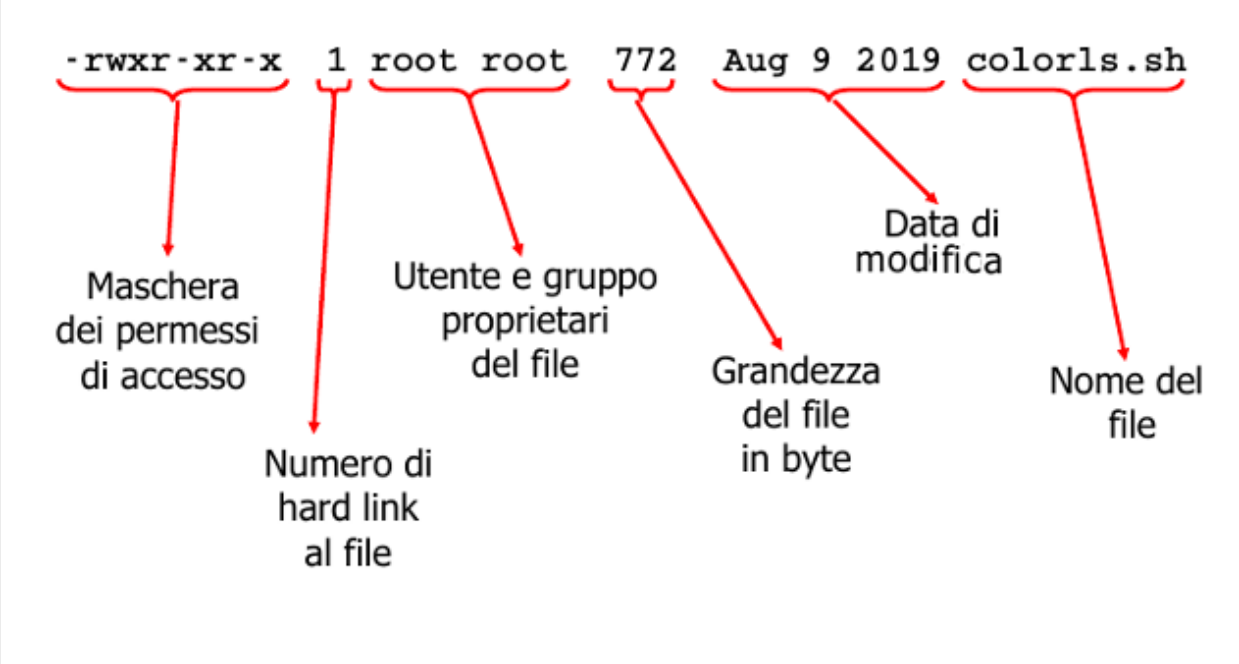
```
ls --help
ls -a -l --color /etc
```





# Il comando `ls`

Per ogni file elencato, l'opzione `-l` elenca una riga della forma:



# Maschera dei permessi

La maschera dei permessi di accesso è composta da 10 caratteri

- il primo carattere indica il tipo di file. I principali tipi sono:
  - `-` file regolare
  - `d` directory
  - `l` link simbolico
  - `s` socket
  - `c`, `b` file relativo a device
- i successivi 9 caratteri sono divisi in 3 triple di caratteri
- le 3 triple indicano i permessi di `owner`, `group`, e `others`
- 3 caratteri di ogni tripla possono essere il carattere `-`, oppure:

  posizione 1: `r`, indica presenza di permesso di lettura
  posizione 2: `w`, indicano presenza di permesso di scrittura
  posizione 3: `x`, indicano presenza di permesso di esecuzione nel caso dei file o attraversamento nel caso delle directory
  (esistono anche altri simboli con uso più specifico...)





# I comandi `pwd` e `cd`

**sintassi** (semplificata): **`pwd`**
Stampa il pathname della present working directory

**sintassi**: **`cd`** `[pathname]`
Modifica la present working directory

- `pathname` è un pathname di una directory, se omesso si usa la home directory
- esempi:

```
user$ cd /home
user$ pwd
/home
user$ cd /usr/lib
user$ pwd
/usr/lib
user$ cd
user$ pwd
/home/user
```

# I comandi `mkdir` e `rmdir`

**sintassi**: **`mkdir`** `pathname`
Crea la directory che ha `pathname` come pathname

- se `pathname` esiste si ha un errore
- l'opzione `-p` causa la creazione di tutte le directory antenate della directory in `pathname`, se non giá presenti
- `pathname` può essere una lista di pathname separati da spazi

**sintassi**: **`rmdir`** `[opzioni] pathname`
Rimuove le directory indicate nella lista pathname

- la/le directory possono essere rimosse solo se sono vuote
- con l'opzione `-p` rimuove di tutti gli antenati specificati nel pathname
  Ad esempio `rmdir -p aa/bb/cc` rimuove (se non hanno ulteriori contenuti) le directory `cc`, `bb`, e `aa`.
  Mentre `rmdir aa/bb/cc` rimuove solo `cc`.





# I comandi `cp`, `mv` e `rm`

**sintassi**: **`cp`** `[opzioni] paths path`

- se `paths` è un singolo pathname, crea una copia con pathname `path`
- se `paths` è una lista di pathname e `path` è una directory, copia tutti gli elementi della lista `paths` in `path`
- l'opzione `-R` permette di copiare ricorsivamente le directory elencate in `paths`
- con l'opzione `-i` chiede conferma prima di sovrascrivere elementi già esistenti in `path`

**sintassi**: **`mv`** `[opzioni] paths path`
Simile a `cp`, ma sposta invece di copiare

**sintassi**: **`rm`** `[opzioni] paths`
Rimuove tutti i file della lista `paths`. Con l'opzione `-R` permette di rimuovere ricorsivamente le directory (anche se NON sono vuote)

# Il comando `chmod`

**sintassi:** **`chmod`** `[opzioni] mode pathname`
Modifica la maschera dei permessi associata a tutti gli elementi della lista pathname

- l'opzione `-R` propaga ricorsivamente l'operazione alle sotto-directory
- `mode` specifica la nuova maschera dei permessi con sintassi simbolica o numerica (in codice ottale)

  **simbolica**: ha la forma `target op permessi`
  - `target` è una stringa di lettere `a u g o`
    (abreviazioni per `all`, `userowner`, `group`, `others`
  - `op` può essere uno tra `= - +`
  - `permessi` può essere una sottostringa di `rwx`





# Il comando `chmod`

**numerica**: ha la forma di una tripla di cifre ottali `NNN`
- ogni cifra ottale N si deve interpretare come una tripla di bit BBB
- i 9 bit corrispondono ai 9 caratteri della maschera dei permessi
- bit 0 significa assenza del permesso, bit 1 presenza del permesso

Esempi:

- la maschera dei permessi `rwxr--rw-` può essere assegnata al file `miofile.txt` tramite il comando
  `chmod 746 miofile.txt`
  ove le cifre ottali 746 corrispondono alle triple di bit 111 100 110
- eseguendo ora il comando
  `chmod g=x miofile.txt` si ottiene la maschera dei permessi
  `rwx--xrw-`

# I comandi `cat`, `more` e `less`

**sintassi:** **`cat`** `[opzioni] pathname`
Scrive sullo stream di output il contenuto dei file elencati nella lista pathname

**sintassi**: **`more`** `pathname`
Visualizza il contenuto dei file elencati nella lista pathname una pagina alla volta. La dimensione della pagina dipende dallo schermo. È possibile effettuare lo scroll di una riga (digitanto invio) o di una pagina (digitanto spazio). Si può anche cercare del testo (digitanto /)

**sintassi**: **`less`** `pathname`
Simile a `more`, ma permette di usare le freccie (up e down) per muoversi avanti e indietro nel testo





# I comandi `grep`, `sort`, `diff`, `wc`

**sintassi:** **`grep`** `[opzioni] stringa pathname`
Scrive sullo stream di output tutte le righe che contengono la `stringa` dei file elencati in pathname

**sintassi**: **`sort`** `[opzioni] path`
Scrive sullo stream di output tutte le righe contenute nel file path in ordine lessicografico

**sintassi**: **`diff`** `[opzioni] path1 path2`
Compara riga per riga i due file indicati da `path1` e `path2` e visualizza le differenze

**sintassi**: **`wc`** `[opzioni] pathname`
Conta il numero di righe, parole, caratteri presenti nei file elencati nella lista `pathname`

# I comandi `w`, `ps`, e `top`

**sintassi:** **`w`** `[opzioni]`
Visualizza gli utenti attualmente presenti e i processi che questi stanno eseguendo

**sintassi**: **`ps`** `[opzioni]`
Visualizza un report sui processi attualmente in esecuzione, limitatamente ai processi con lo stesso (effective) UID di chi esegue il comando. L'opzione `-l` fornisce un output dettagliato. L'opzione `-A` visualizza tutti i processi.

**sintassi**: **`top`** `[opzioni]`
Visualizza in modo dinamico dettagli sui processi in esecuzione (fino a che non si esce digitanto `q`)





# I comandi `basename`, `dirname`, `which`, e `history`

**sintassi:** **`basename`** `[opzioni] pathname`
**sintassi:** **`dirname`** `[opzioni] pathname`
Manipolano la stringa `pathname` restituendo rispettivamente l'ultima componente del pathname o il pathname senza l'ultima componente. Ad esempio:

```
user$ basename /home/user/dir1/dir2/file.txt
file.txt
user$ dirname bin/dirA/dirB/file.txt
bin/dirA/dirB
```

**sintassi**: **`which`** `comando`
Visualizza il pathname del comando esterno (ovvero il path del file che la bash eseguirebbe se si invocasse comando)

**sintassi**: **`history`**
Comando (solitamente builtin) che visualizza l'elenco degli ultimi comandi eseguiti

# I comandi `echo`, `clear` e `sleep`

**sintassi:** **`echo`** `[opzioni] stringa`
Stampa nello stream di output la stringa. È possibile accedere ai valori delle variabili della bash (le vedremo meglio in seguito...)
Un esempio che usa le variabili `$HOME`, `$PATH` e `$PWD` :

```
user$ echo "una stringa"
una stringa
user$ echo "la mia home directory: $HOME"
la mia home directory: /home/user
user$ echo "la directory corrente: $PWD"
la directory corrente: /home/user
user$ echo "valore della variabile: $PATH"
valore della variabile: /bin:/usr/bin:/home/user/bin
```

**sintassi**: **`clear`**
Cancella lo schermo

**sintassi**: **`sleep`** `numero`
Effettua una attesa di `numero` secondi





# I comandi `bash` e `exit`

**sintassi:** **`bash`** `[opzioni]`
Esegue una shell `bash` figlia della bash attiva
**sintassi**: **`exit`**
Causa la terminazione della shell in cui viene eseguito

```
user$ ps
  PID TTY          TIME CMD
 1715 pts/14   00:00:00 bash
 3746 pts/14   00:00:00 ps
user$ bash
user$ ps
  PID TTY          TIME CMD
 1715 pts/14   00:00:00 bash
 3750 pts/14   00:00:00 bash
 3762 pts/14   00:00:00 ps
user$ exit
exit
user$ ps
  PID TTY          TIME CMD
 1715 pts/14   00:00:00 bash
 3769 pts/14   00:00:00 ps
```

# Esercitazione (1)

1. Scoprite quale è la vostra directory corrente [comando `pwd`] (e ricordatevela...)
2. Visualizzate il contenuto della directory corrente [comando `ls`]
3. Scoprite (usando il comando `man`) come visualizzare il contenuto della directory corrente in modo che i file siano ordinati rispetto alla data di modifica. E poi in ordine inverso.
4. Visualizzate il contenuto delle tre directory `/`, `/usr/` e `/home/` con tre comandi diversi e poi con un unico comando
5. Tramite il comando `cd` cambiate la propria directory corrente in `/dev/`
   Verificate con `pwd` l'effetto del comando precedente e visualizzate contenuto della directory corrente [comando `ls -l`]
   Verificate ora la presenza in `/dev/` di file regolari, di directory e di file speciali
6. Eseguite il comando `whatis mkdir`
   Fatto ciò tramite il `man` scoprite come si usa il comando `mkdir`
7. Assicuratevi di essere nella propria home directory [comando `cd` senza parametri, e poi verificate con `pwd`] e create una nuova directory di nome `fiori`





# Esercitazione (2)

8. Spostatevi nella directory `fiori` [comando `cd`] e visualizzatene il contenuto [comando `ls -al`]
9. Restando nella directory `fiori` creare una sotto-directory `fioribelli` ed una sotto-directory di `fioribelli` chiamata `fiorirossi`
10. Scoprite (con il `man`) cosa fa il comando `tree` ed eseguitelo nella directory `fiori` (se nel vostro sistema il comando `tree` non è installato, provate ad eseguire `ls -alR`)
11. Scoprite (con il `man`) cosa fa il comando `touch`
12. Usando il comando `touch`, create un file `viola.txt` nella directory `fiori` [comando `touch viola.txt`]
    Poi create un file `tulipano.txt` nella directory `fioribelli`
    Poi create un file `rosa.txt` nella directory `fioribelli/fiorirossi`
13. Restando nella directory `fiori`, ripetete il comando `tree` Che differenze notate rispetto a prima?
14. Cambiate la directory corrente in `fiori/fioribelli/fiorirossi` e visualizzatene il contenuto
15. Quale è il comando per copiare un file e come si usa? [suggerimento: `man cp`]

# Esercitazione (3)

16. Ora copiate il file `rosa.txt` nella propria home directory. Ricordate quale è? Sì? Verificate se avete buona memoria eseguendo il comando `echo $HOME`
17. Riuscite a prevedere cosa fanno i seguenti comandi? (verificate!)
```
echo "La mia home: $HOME La dir corrente: $PWD"
echo "La mia home: $HOME \n La dir corrente: $PWD"
echo -e "La mia home: $HOME \n La dir corrente: $PWD"
```
18. Usando i pathname assoluti e relativi ed il comando `cd` navigate nel filesystem cercando di capire come è strutturato, quanti e quali nodi sono contenuti in `/` e nelle sue sotto-cartelle
19. Eseguite il comando `ps`
20. Verificate cosa fa il comando `ps` invocato con le opzioni `-A` e/o `-l`
21. Eseguite ora i tre comandi `ps -A` , `ps -l` e `ps -Alf`
22. Scoprite usando il `man` cosa fa il comando `top` e in particolare come si esce da esso una volta lanciato.
23. Eseguite ora il comando `top` e poi... uscite
24. Riposizionatevi nella directory `fiori` [comando `cd`] e create una directory `elimina`





# Esercitazione (4)

25. Eseguite `man mv` per scoprire tutto sul comando `mv`
26. Ora spostate la directory `fioribelli` e tutto il suo contenuto nella directory `elimina`
27. Rimuovete il file `elimina/fioribelli/fiorirossi/rosa.txt`
    (per verificare come si fa: comando `man rm`)
    Provate a rimuovere il file `jasmine.txt` (che non esiste...)
28. Scoprite cosa fa `rmdir` e che differenze ci sono con `rm`
    Scoprite cosa fa l'opzione `-r` di `rm`. E l'opzione `-R`?
29. Con i comandi `rm` e `rmdir` cancellare `elimina` e tutto il suo contenuto
30. Scoprire cosa fa il comando `ln` ed a cosa serve l'opzione `-s`
31. Create un file `temp.txt` [comando `touch`]
32. Create un link simbolico di nome `tempfinto.txt` che punti a `temp.txt`
33. Create un altro link simbolico di nome `altrofinto.txt` che punti sempre a `temp.txt`
34. Eseguite ora il comando `ls -al`
35. Rimuovete il link `tempfinto.txt`
36. Eseguite ora il comando `ls -al`
37. Rimuovete il vero file `temp.txt`

# Esercitazione (5)

38. Eseguite ora il comando `ls -al` (notate qualcosa di “nuovo”?)
39. Rimuovete il link `altrofinto.txt`
40. Usando il manuale `man` documentatevi sui comandi `more` , `less` , `grep`
41. Documentatevi sui comandi `df` , `du`
42. Documentatevi sul comando `history` , usatelo per scoprire gli ultimi comandi che avete eseguito e scoprite come richiedere l’esecuzione di uno di essi senza digitarlo nuovamente
43. Create un file (usando il comando `touch miofile`) e visualizzate la maschera dei permessi
44. Documentatevi sul comando `chmod`
45. Con il comando `chmod` cambiate la maschera dei permessi di `miofile` Usando la sintassi simbolica e poi visualizzate i nuovi permessi
46. Usate ancora `chmod` per cambiare nuovamente i permessi di `miofile`, ma con la sintassi numerica e poi visualizzate i nuovi permessi
47. Cancellate il file `miofile`
48. Documentatevi sui comandi `chown` , `chgrp`
49. Consultate il `man` per scoprire più dettagli su tutti i comandi visti a lezione