

# INFORMATICA

Intelligenza Artificiale & Data Analytics

Precorsi a.a. 2023/24

Docente: Pietropolli Gloria

# 4. LINUX 101

Installazione di Ubuntu su *macchina virtuale*

# TIPI DI INSTALLAZIONE DI UBUNTU

Installazione come unico OS

Installiamo Ubuntu su una macchina che non ha nessun altro sistema operativo installato.

Installazione in *dual boot*

Installiamo Ubuntu su una macchina che ha già un altro OS installato. Dobbiamo effettuare una **partizione** di un disco fisso (oppure usarne un altro vuoto) e installando lì Ubuntu. All'avvio del PC, dovremo selezionare quale OS vogliamo far partire.

Installazione su macchina virtuale

Esistono dei programmi (es. [VirtualBox](#)) che consentono di dedicare uno spazio disco all'*emulazione* di un computer fisico (*macchina virtuale*) su cui installare un OS secondario, che gira in dipendenza dell'OS principale su cui è installato l'*emulatore*.

Utilizzo del Sottosistema a Linux Win10

Win 10 permette agli utenti esperti di installare dei *plug-in* per permettere di girare software Linux direttamente in ambiente Windows senza l'utilizzo di dual boot o macchine virtuali. Attualmente non offre compatibilità integrale rispetto a Ubuntu.

# INSTALLAZIONE COME MACCHINA VIRTUALE – FASI PRELIMINARI

Per installare Ubuntu come macchina virtuale, abbiamo innanzitutto bisogno di installare [VirtualBox](#) e scaricare un'immagine disco di [Ubuntu 20.04](#).

L'immagine disco che andiamo a scaricare simula la struttura dei dati come se fosse questi fossero contenuti all'interno di un DVD.

Vogliamo fare in modo di *emulare* il processo secondo cui installiamo il sistema operativo inserendo il DVD nel lettore, senza però avere bisogno di un DVD fisico!



# INSTALLAZIONE COME MACCHINA VIRTUALE - FASI PRELIMINARI

## 1. Installare VirtualBox

- Vai al sito ufficiale: [virtualbox.org](http://virtualbox.org).
- Scarica la versione adatta al tuo sistema operativo (Windows, macOS, Linux).
- Esegui il file di installazione e segui le istruzioni guidate.



# INSTALLAZIONE COME MACCHINA VIRTUALE – FASI PRELIMINARI

## 2. Scaricare un sistema operativo (ISO)

- Per creare una VM, hai bisogno di un file **ISO** del sistema operativo che desideri installare (es. Ubuntu, Windows).
- Vai al sito ufficiale del sistema operativo e scarica l'ISO. Esempi:
  - **Ubuntu:** [ubuntu.com/download](http://ubuntu.com/download)
  - **Windows:** [microsoft.com](http://microsoft.com)



# INSTALLAZIONE COME MACCHINA VIRTUALE - FASI PRELIMINARI

## 3. Creare una nuova VM in VirtualBox

- Apri **VirtualBox** e clicca su "**Nuova**" per creare una nuova macchina virtuale.
- **Assegna un nome** alla VM e scegli il sistema operativo (es. Linux, Windows).
- Seleziona la quantità di **RAM** da dedicare alla VM (consigliato almeno 2GB per Linux, 4GB per Windows).



# INSTALLAZIONE COME MACCHINA VIRTUALE - FASI PRELIMINARI

## 4. Collegare il file ISO

- Durante la creazione, seleziona "**Crea un nuovo disco virtuale**".
- Nel menu successivo, **collega il file ISO** scaricato per installare il sistema operativo.
- Clicca su "**Avvia**" per avviare la VM.



# INSTALLAZIONE COME MACCHINA VIRTUALE - FASI PRELIMINARI

## 5. Installare il sistema operativo

- La VM si avvierà dal file ISO.
- Segui le istruzioni dell'installazione del sistema operativo (come faresti su un PC fisico).

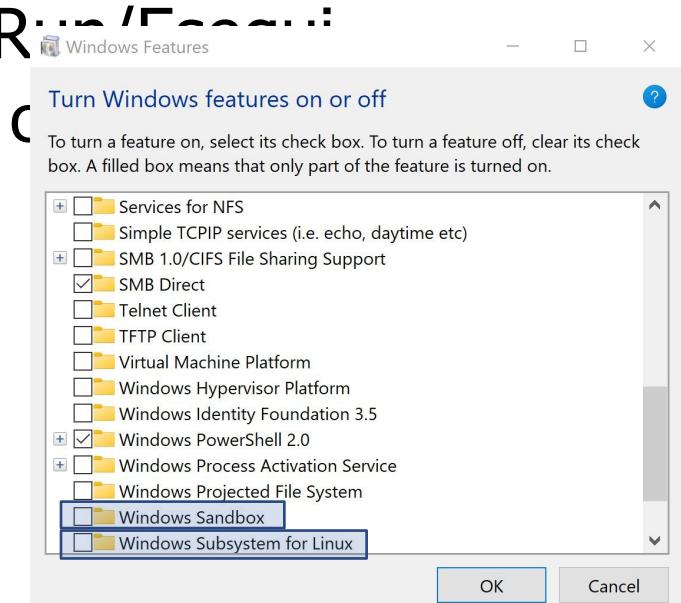


# POSSIBILI FONTI DI INCOMPATIBILITÀ

VirtualBox non funziona se su Win10 è già attivo il Linux Subsystem e se sono attive altre funzionalità come HyperV e Windows Sandbox.

Per disattivare questi servizi, attivare la finestra `Ricerca` premendo la combinazione di tasti +R e digitare `Turn Windows features on or off`. Cercare i 3 nomi sopra indicati e togliere il tick.

Riavviare il computer.



# COSA ACCADE ORA?

Apriamo VirtualBox e  
selezioniamo  New  
Segue lezione pratica



# 4. LINUX 101

Interfaccia *desktop*

# L'INTERFACCIA DESKTOP DI UBUNTU

File explorer di Ubuntu (si chiama *Nautilus*)

Barra applicazioni (analogia a Windows 10)

Start menu (analogo a Windows 10)



Interfaccia

# 4. LINUX 101

Interfaccia a linea di comando

# RIASSUNTO DALLA LEZIONE 2

## L'INTERFACCIA GRAFICA

Tuttavia, i computer dell'epoca non supportavano finestre grafiche intuitive per un pubblico generale.

Inizialmente, non esistevano le interfacce grafiche che ci permettevano di visualizzare graficamente le cartelle con icone contenenti file e programmi.



# BREVE STORIA DELL'INTERAZIONE UOMO-MACCHINA (I)

I primi computer non avevano né schermo né tastiera: l'I/O avveniva tramite schede perforate.

Successivamente si integrò nel sistema la **telescrivente**, un dispositivo elettromeccanico per trasmettere messaggi telegrafici, esistente già da fine '800.

Simile a una macchina da scrivere, il ruolo del monitor è svolto dalla carta stampata.

Si sviluppa in questo contesto il paradigma dell'interfaccia a **linea di comando**.



# BREVE STORIA DELL'INTERAZIONE UOMO-MACCHINA (I)

Appena ad inizio anni '60 si diffonde l'utilizzo dei monitor.

Interazione più diretta con il computer:

- Il computer può mostrare i risultati delle sue elaborazioni su schermo
- L'umano può digitare istruzioni tramite la tastiera e vedere ciò che sta digitando sul monitor

Il monitor sostituisce il ruolo della carta della telescrivente: permette di vedere uno storico dei comandi recenti (e del rispettivo output del computer)

- Non è possibile quindi «modificare» la storia pregressa

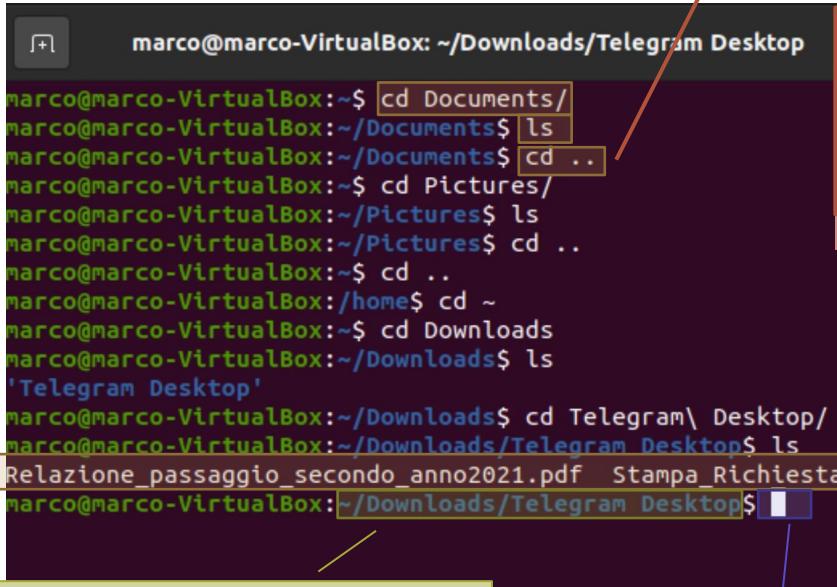


INTERFACCIA A LINEA  
DI COMANDO

# IL TERMINALE A LINEA DI COMANDO

È un'interfaccia che utilizza solamente **tastiera e schermo (no mouse!)** e che permette di lavorare con una grandissima parte delle funzionalità base dell'OS

Comandi per il sistema operativo (specificamente per la navigazione del filesystem)



The screenshot shows a terminal window with the following session:

```
marco@marco-VirtualBox: ~/Downloads/Telegram Desktop
marco@marco-VirtualBox:~$ cd Documents/
marco@marco-VirtualBox:~/Documents$ ls
marco@marco-VirtualBox:~/Documents$ cd ..
marco@marco-VirtualBox:~/Pictures$ ls
marco@marco-VirtualBox:~/Pictures$ cd ..
marco@marco-VirtualBox:~$ cd ..
marco@marco-VirtualBox:/home$ cd ~
marco@marco-VirtualBox:~$ cd Downloads
marco@marco-VirtualBox:~/Downloads$ ls
'Telegram Desktop'
marco@marco-VirtualBox:~/Downloads$ cd Telegram\ Desktop/
marco@marco-VirtualBox:~/Downloads/Telegram Desktop$ ls
Relazione_passaggio_secondo_anno2021.pdf Stampa_Richiesta_Misso...
```

Annotations:

- Cartella corrente = *working directory*** → se eseguo comandi per filesystem, li esegue in questa cartella
- Riga corrente** → attesa comando utente
- Output dell'OS: composizione della cartella indicata**

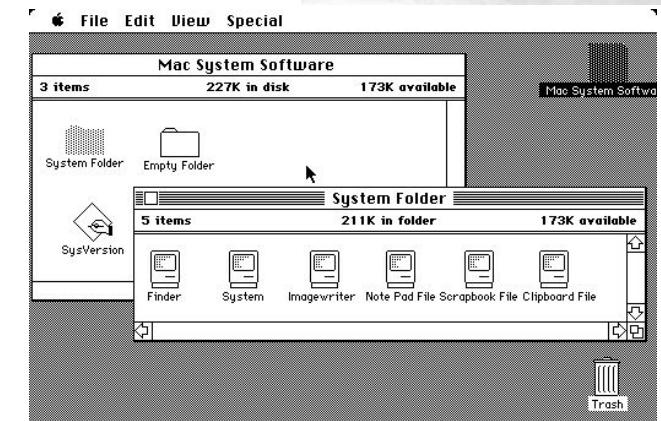
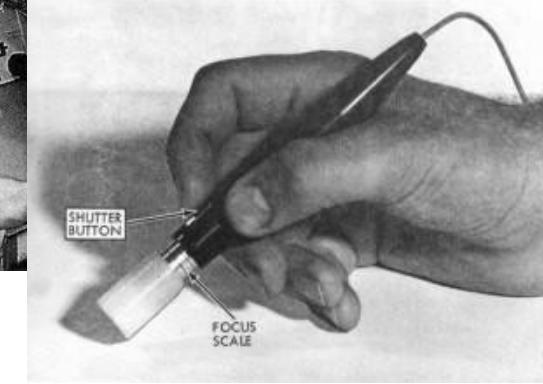
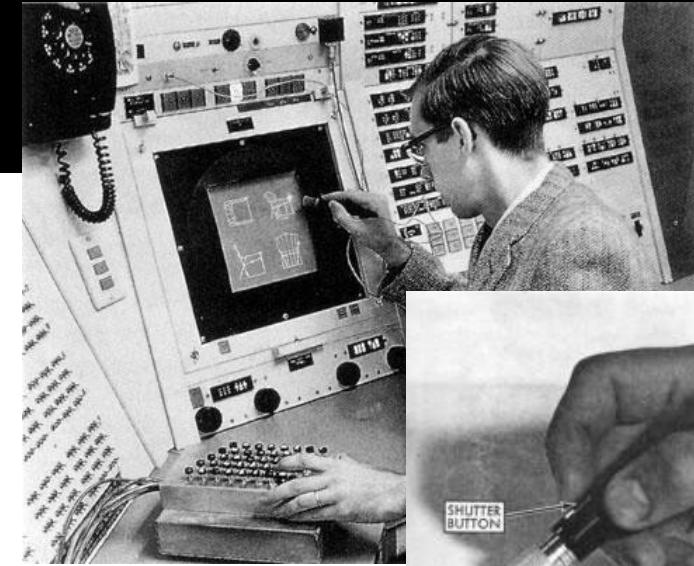
# BREVE STORIA DELL'INTERAZIONE UOMO-MACCHINA

I mouse arrivarono appena a fine anni '60

- Prima si usavano i tasti ↑←→Home End

La prima interfaccia grafica nacque nel '63,  
utilizzava una penna grafica, ma rimase  
solamente un prototipo

La prima interfaccia grafica *desktop* fu  
sviluppata a partire da '73 dalla Xerox e fu poi  
copiata, prima da Apple e poi da Microsoft



# PERCHÉ, NONOSTANTE TUTTO, SI CONTINUA AD UTILIZZARE L'INTERFACCIA A LINEA DI COMANDO?

Velocità di utilizzo

Accesso rapido a funzionalità secondarie, senza dover passare per menu di impostazione multipli (es. Pannello di controllo)

ma soprattutto...

L'utilizzo di comandi testuali favorisce l'automazione dei compiti.  
È difficile automatizzare un compito (es. con un programma) affinché vada ad eseguire delle istruzioni direttamente su interfaccia grafica.

Andando invece ad integrare i comandi del terminale nel proprio codice, si riesce ad interagire direttamente con l'OS, anche se il linguaggio che si sta utilizzando non prevede tali funzionalità

# BASH - PRACTICUM



1

- Shell e bash

2

- Comandi per navigazione filesystem

3

- Comandi per manipolazione filesystem

4

- Altri comandi miscellanei

5

- Package manager per Ubuntu

# CHE COS'È UNA SHELL? E BASH?

Shell → «componente fondamentale di un sistema operativo che permette all'utente il più alto livello di interazione con lo stesso»

*che cosa significa?*

L'interazione avviene tramite stringhe di testo

Bash è una shell testuale per Linux (e presente anche sui sistemi Mac). È derivata direttamente dalla shell di Unix

Derivazione del termine «shell»: è il guscio, la parte dell'OS che è visibile all'utente finale.

# COME OTTENERE UN TERMINALE CON BASH?



- Installare WSL, che porta con sé una shell di Ubuntu  
Oppure
- Installare CygWin o Git for Windows, che installano a loro volta una shell che supporta Bash



MacOS ha pre-installato un terminale che supporta una buona parte delle funzionalità Bash

Cercare ed eseguire terminal.app

Per avere tutte le funzionalità e poter installare i programmi che usualmente girano su Linux, è consigliata l'installazione di [Homebrew](#).

# INOSTRI PRIMI COMANDI BASH: COMANDI FILESYSTEM - PWD

Inizieremo ad utilizzare Bash per dare alcune istruzioni per la navigazione e la manipolazione del filesystem.

Attraverso questi comandi, impareremo la struttura di un filesystem in Linux.

Il primo comando che utilizziamo è **pwd**

Print Working Directory

```
marco@marco-VirtualBox:~/Downloads/Telegram Desktop$ pwd  
/home/marco/Downloads/Telegram Desktop
```

«User home folder», anche indicata con la tilde «~»

Percorso (path)

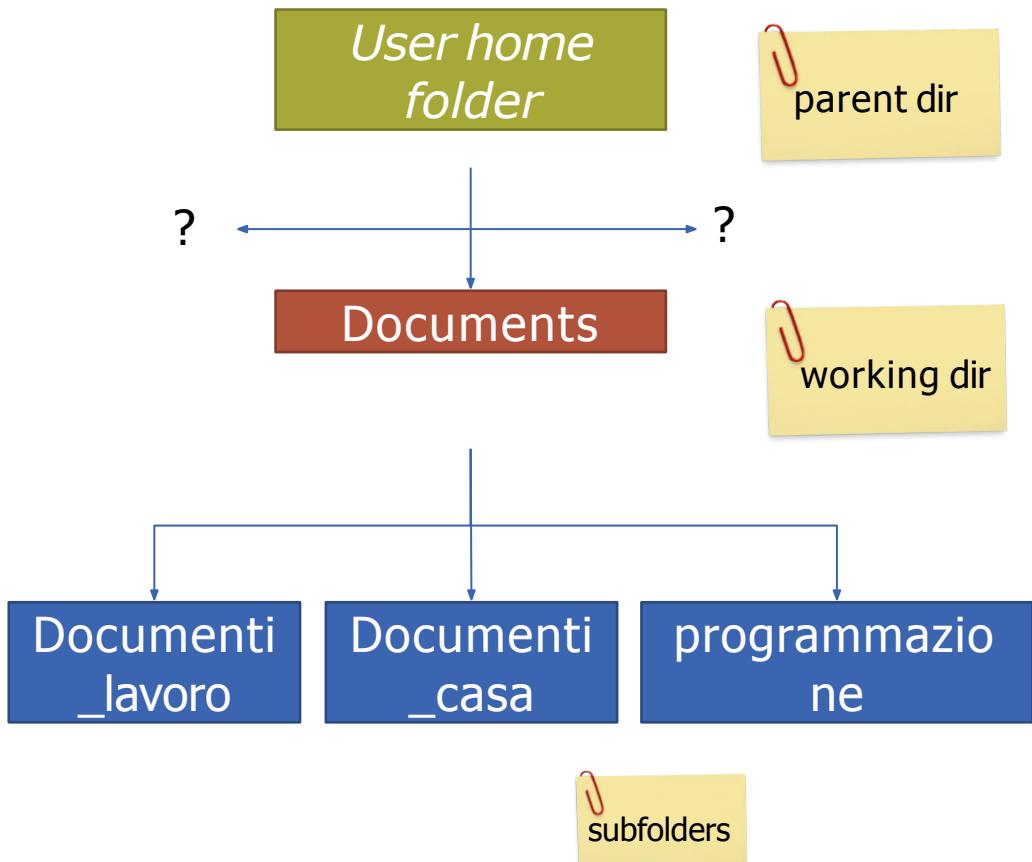
Posizione della working directory all'interno del sistema *ad albero* del filesystem

# CD – CHANGE DIRECTORY

**cd:** cambiare *working directory*

Richiede il passaggio di un **argomento**  
→ **nome della cartella di destinazione**

Dobbiamo conoscere la struttura della working directory per muoverci.



# CD - IN PRATICA

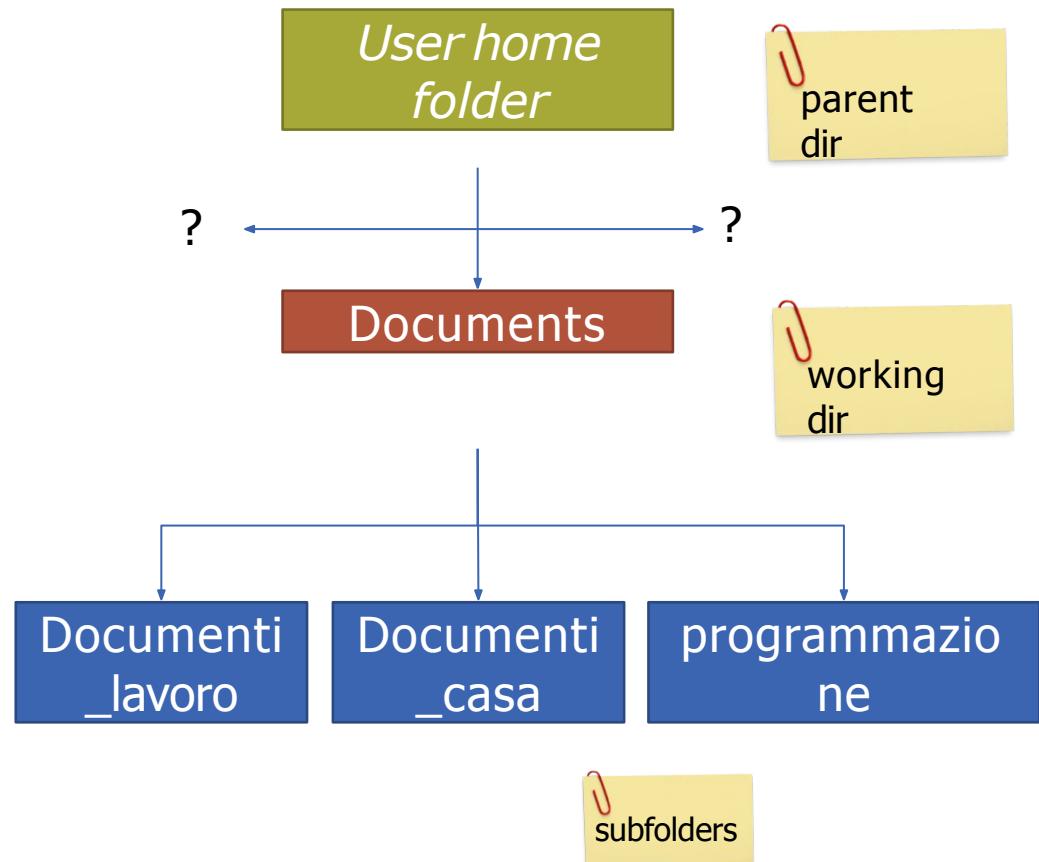
Proviamo a muoverci in una delle sottocartelle - programmazione

**cd programmazione**

Vogliamo tornare indietro in - Documents

**cd Documents**

Che cosa succede?



# CD – PARENT DIRECTORY

Il parametro dopo cd si riferisce alle **sottocartelle**

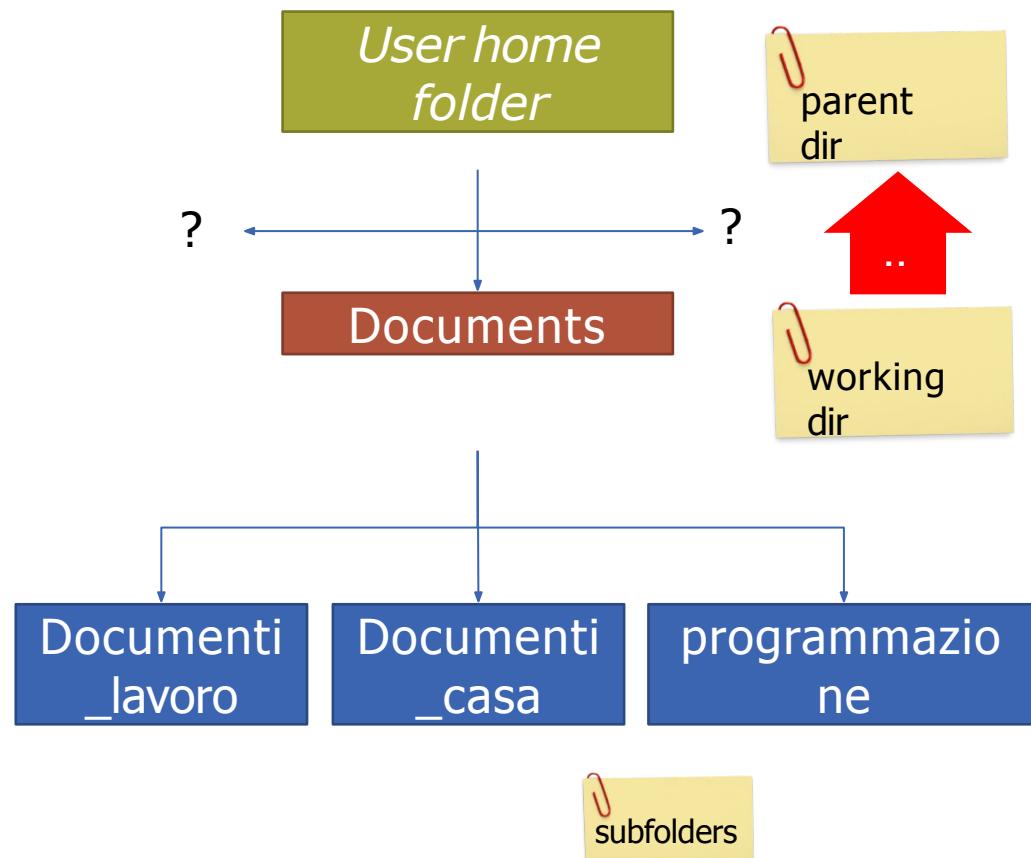
→ riceviamo un errore

Per navigare all'indietro possiamo usare la stringa “..”

→ si riferisce sempre alla *parent dir* della *working dir*

`cd ..`

ci permette di tornare alla dir - Documents



# CD – ARGOMENTI COMPOSITI

Come argomento di cd possiamo anche indicare una **catena di sottocartelle** o di parent dir separati dallo slash «/»

Esempi

- **cd Documents/programmazione**
- **cd ../../**

Che cosa fa questo comando di preciso?

# CD – PATH ASSOLUTI

È anche possibile indicare, come argomento di cd, un **percorso assoluto**

Ricordiamo da prima:

```
marco@marco-VirtualBox:~/Downloads/Telegram Desktop$ pwd  
/home/marco/Downloads/Telegram Desktop
```

Percorso (path) assoluto

Posizione della *working directory* all'interno del sistema ad albero del filesystem

Possiamo quindi digitare

```
cd /home/marco/Downloads/Telegram Desktop
```

Che cosa succede?

# BASH – CARATTERI RISERVATI

- • - Indica la directory superiore (parent directory).

Esempio: cd .. ti porta alla directory che contiene quella corrente.

- - Rappresenta la directory corrente.

Esempio: ./script.sh esegue lo script script.sh nella directory corrente.

- / - Usato per separare le directory in un percorso.

Esempio: /home/utente rappresenta il percorso assoluto della directory utente.

- \* - Wildcard, rappresenta qualsiasi serie di caratteri.

Esempio: ls \*.txt mostra tutti i file con estensione .txt.

- ? - Rappresenta un singolo carattere qualsiasi.

Esempio: ls file?.txt elenca file come file1.txt, fileA.txt, ma non file12.txt.

# BASH – ESCAPE CHARACTER

Se vogliamo utilizzare cd per dirigerci in un percorso in cui un file o una cartella hanno spazi nel nome, dobbiamo utilizzare il carattere «\», il *backslash*.

\

backslash

Utilizzato per indicare il **significato letterario** del simbolo successivo, e non il suo significato di Bash

/

slash

Utilizzato per separare i nomi di cartelle e file nei percorsi

```
cd /home/marco/Downloads/Telegram\ Desktop
```

# LS - LIST FILES

**ls** ci permette di ottenere in output la lista dei file presenti in una determinata cartella.

Può funzionare sia con che senza **argomento**

ls

Lista file cartella corrente

ls percorso/cartella

Lista file cartella indicata

equivalente a

ls .

```
marco@marco-VirtualBox:~/Documents$ ls
Documenti_casa    documento1.doc  programmazione
Documenti_lavoro  liste_spesa.txt

marco@marco-VirtualBox:~/Documents$ ls /home/marco/Documents/programmazione/
array.py  helloworld.c  hello_world.py
marco@marco-VirtualBox:~/Documents$ ls programmazione
array.py  helloworld.c  hello_world.py
```

Perché sono uguali?

# BASH – PARAMETRI OPZIONALI

Alcuni comandi ammettono anche **parametri opzionali**

I parametri opzionali vanno indicati con - o --

**ls Documents**

**-a**

Flag corto

**ls Documents**

**--all**

Flag esteso

Possiamo anche visualizzare l'intero albero filesystem che si trova nella cartella in cui ci troviamo:

**ls Documents --R**

# MKDIR – CREAZIONE CARTELLE

**mkdir** (*make directory*) creare una cartella vuota

→ **parametro**: percorso (relativo/assoluto) della cartella + nome

La cartella di lavoro è ~/Documents/programmazione e vogliamo creare una cartella chiamata precorsi\_informatica in questo percorso

**mkdir precorsi\_informatica**

**mkdir ~/Documents/programmazione/precorsi\_informatica**

```
marco@marco-VirtualBox:~/Documents/programmazione$ mkdir precorsi_informatica
marco@marco-VirtualBox:~/Documents/programmazione$ ls
array.py  helloworld.c  hello_world.py  precorsi_informatica
```

# TOUCH - CREAZIONE FILE VUOTI

Creare un file vuoto comandi\_bash.txt dentro della cartella creata  
**touch:** creazione di nuovi file

→ **parametro:** percorso (assoluto/relativo) del file + nome

Tramite «» posso concatenare più comandi da eseguire in sequenza

## Esercizio

Provate a proporre i comandi per la creazione del file.  
NB: la cartella di lavoro è ancora programmazione

cd precorsi\_informatica; touch comandi\_bash.txt

touch precorsi\_informatica/comandi\_bash.txt

touch ~/Documents/precorsi\_informatica/comandi\_bash.txt

# ESERCIZIO INSIEME

## 1. Navigazione e creazione della cartella:

- Usa i comandi che conosci per visualizzare in quale directory ti trovi attualmente.
- Spostati sul Desktop (se non sei già lì).
- Crea una nuova cartella chiamata ProgettoTerminale.

## 2. Navigazione e gestione dei file:

- Entra nella cartella ProgettoTerminale.
- All'interno, crea un'altra cartella chiamata Documenti.
- Crea tre file vuoti nella cartella Documenti con i nomi appunti.txt, compiti.txt e bozza.txt.

## 3. Navigazione e manipolazione con caratteri speciali:

- Torna indietro nella cartella principale ProgettoTerminale senza utilizzare il percorso assoluto.
- Crea una nuova cartella chiamata Immagini.
- Crea due file vuoti con i nomi foto1.png e foto2.png nella cartella Immagini usando caratteri speciali per evitare di digitare l'intero nome del file due volte.

## 4. Lista e verifica:

- Visualizza la lista completa dei file all'interno della cartella ProgettoTerminale e delle sue sottocartelle, assicurandoti che tutto sia stato creato correttamente.
- Usa un carattere speciale per visualizzare solo i file .txt presenti in Documenti.

# CP – COPIA FILE

**cp** ci permette di copiare file da una destinazione ad un'altra

A differenza dei precedenti comandi, richiede due parametri

1.

File da copiare

2.

Destinazione della copia

La cartella di lavoro è precorsi\_informatica

Vogliamo creare una copia di comandi\_bash.txt  
all'interno della cartella superiore, programmazione

```
cp comandi_bash.txt ~/Documents/programmazione
```

# CP – DESTINAZIONE CON PERCORSI RELATIVI

## Esercizio

L'esempio qui sopra contiene la destinazione con percorso assoluto. Come possiamo ripetere il comando usando solo percorsi relativi?

Nota: working dir = precorsi\_informatica

```
cp comandi_bash.txt ..
```

# CP – RINOMINAZIONE DEL FILE

Ammettiamo di voler rinominare il file da copiare, da comandi\_bash.txt a bash\_esempi.txt

Possiamo copiare e rinominare in un solo comando grazie a cp

Basta accodare al percorso di destinazione il nome del file

```
cp comandi_bash.txt ~/Documents/programmazione/bash_esempi.txt
```

```
cp comandi_bash.txt ../bash_esempi.txt
```

# NOTA – ESTENSIONE DEL FILE

Nei sistemi Unix, è anche possibile creare dei file senza estensione Es.

```
cp comandi_bash.txt .../bash_esempi
```

Ho appena creato un file chiamato bash\_esempi e senza alcuna estensione

In Bash, esiste il comando file, che tenta di restituire il tipo del file anche se quest'ultimo è sprovvisto di estensione

```
marco@marco-VirtualBox:~/Documents/programmazione$ file bash_esempi
bash_esempi: UTF-8 Unicode text
```

# CP - COPIA NELLA STESSA CARTELLA

Ammattiamo ora di voler creare una copia di comandi\_bash.txt all'interno di programmazione, in modo da poterla modificare tenendo una copia dell'originale.

Come faccio ad indicare la cartella stessa in un percorso relativo?

Ricordiamoci di alcuni caratteri riservati...



Cartella corrente



Cartella parent

cp comandi\_bash.txt .

Tutto OK?

```
marco@marco-VirtualBox:~/Documents/programmazione$ cp comandi_bash.txt .
cp: 'comandi_bash.txt' and './comandi_bash.txt' are the same file
```

# CP - COPIA NELLA STESSA CARTELLA – RINOMINAZIONE

L'errore è relativo al fatto che esiste già, all'interno di precorsi\_informatica, un file con lo stesso nome

Se vogliamo copiare un file all'interno della stessa cartella,  
dobbiamo obbligatoriamente rinominarlo

cp comandi\_bash.txt ./comandi\_bash\_copia.txt

cp comandi\_bash.txt comandi\_bash\_copia.txt

```
marco@marco-VirtualBox:~/Documents/programmazione/precorsi_informatica$ cp comandi_bash.txt comandi_bash_copia.txt
marco@marco-VirtualBox:~/Documents/programmazione/precorsi_informatica$ ls
comandi_bash_copia.txt  comandi_bash.txt
```

# CP – COPIA CARTELLE

Ammettiamo ora di voler creare una copia della cartella `precorsi_informatica`, sempre all'interno di programmazione. La chiamiamo `precorsi_informatica_copia`.

`cd ..; cp precorsi_informatica precorsi_informatica_copia`

```
marco@marco-VirtualBox:~/Documents/programmazione/precorsi_informatica$ cd ..
marco@marco-VirtualBox:~/Documents/programmazione$ cp precorsi_informatica/ precorsi_informatica_copia
cp: -r not specified; omitting directory 'precorsi_informatica/'
```

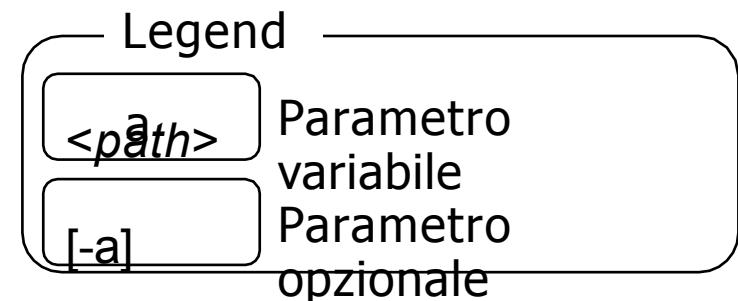
cp di default non permette la copia di cartelle intere, devo specificare il flag `-r` (**recursive**)

Copia tutti i file contenuti nelle sottocartelle dirette, poi tutti quelli delle rispettive sottocartelle, ecc.

`cp -r precorsi_informatica precorsi_informatica_copia`

# RIASSUNTO DEI COMANDI FILESYSTEM

cd	<i>Change Directory</i>	cd <path>
ls	<i>List files</i>	ls [-a -l -t] [<path>]
pwd	<i>Print Working Dir</i>	pwd
touch	<i>Create file</i>	touch <path(s)>
mkdir	<i>Make directory</i>	mkdir <path(s)>
cp	<i>Copy</i>	cp [-r] origin destination



«Navigazione  
filesystem»

«Manipolazione  
filesystem»

# MV – MUOVI FILE

Finora abbiamo visto esempi di copia

Possono esserci casi in cui non voglio copiare un file, ma solo spostarlo altrove

In questo caso posso usare mv, il cui utilizzo è praticamente identico a cp

Muoviamo il file liste\_spesa.txt, contenuto in Documents, nella cartella Documenti\_casa

```
mv ~/Documents/liste_spesa.txt ~/Documents/Documenti_casa
```

```
marco@marco-VirtualBox:~/Documents$ mv liste_spesa.txt Documenti_casa
marco@marco-VirtualBox:~/Documents$ ls Documenti_casa/
liste_spesa.txt
marco@marco-VirtualBox:~/Documents$ ls .
Documenti_casa  Documenti_lavoro  documento1.doc  programmazione
```

# MV – MUOVI CARTELLA

Possiamo parimenti usare mv per muovere intere cartelle

Stavolta, non c'è necessità di usare il flag –r

Esempio: spostare precorsi\_informatica\_copia all'interno della cartella precorsi\_informatica

```
marco@marco-VirtualBox:~/Documents/programmazione$ ls
array.py      helloworld.c  precorsi_informatica
comandi_bash.txt  hello_world.py  precorsi_informatica_copia
marco@marco-VirtualBox:~/Documents/programmazione$ mv precorsi_informatica_copi
a/ precorsi_informatica
marco@marco-VirtualBox:~/Documents/programmazione$ ls
array.py  comandi_bash.txt  helloworld.c  hello_world.py  precorsi_informatica
marco@marco-VirtualBox:~/Documents/programmazione$ ls precorsi_informatica/
comandi_bash_copia.txt  comandi_bash.txt  precorsi_informatica_copia
```

# MV PER RINOMINARE FILE E CARTELLE

Possiamo usare mv per rinominare un file

Esempio: vogliamo rinominare il file liste\_spesa.txt in ListeSpesa.txt

mv liste\_spesa.txt ListeSpesa.txt

```
marco@marco-VirtualBox:~/Documents/Documenti_casa$ ls
liste_spesa.txt
marco@marco-VirtualBox:~/Documents/Documenti_casa$ mv liste_spesa.txt ListeSpes
a.txt
marco@marco-VirtualBox:~/Documents/Documenti_casa$ ls
ListeSpesa.txt
marco@marco-VirtualBox:~/Documents/Documenti_casa$ █
```

Possiamo fare lo stesso con le cartelle

# RM – RIMUOVI FILE O CARTELLA

Il comando rm può essere usato per rimuovere file o cartelle



rm rimuove i file definitivamente, non vengono mandati al cestino!



Anche  
rmdir  
se  
vuota

rm comandi\_bash\_copia.txt  
rm –r precorsi\_informatica\_copia

```
marco@marco-VirtualBox:~/Documents/programmazione/precorsi_informatica$ rm comandi_bash_copia.txt
marco@marco-VirtualBox:~/Documents/programmazione/precorsi_informatica$ rm precorsi_informatica_copia/
rm: cannot remove 'precorsi_informatica_copia/': Is a directory
marco@marco-VirtualBox:~/Documents/programmazione/precorsi_informatica$ rm precorsi_informatica_copia/ -r
marco@marco-VirtualBox:~/Documents/programmazione/precorsi_informatica$ ls comandi_bash.txt
```

# RIMUOVERE FILE O CARTELLE DI SISTEMA

Possiamo rimuovere file o cartelle di sistema com rm o rmdir?

```
marco@marco-VirtualBox:~/Pictures$ rmdir /root
rmdir: failed to remove '/root': Permission denied
marco@marco-VirtualBox:~/Pictures$ rmdir /etc
rmdir: failed to remove '/etc': Permission denied
marco@marco-VirtualBox:~/Pictures$ rmdir /home
rmdir: failed to remove '/home': Permission denied
```

L'utente corrente può modificare solamente file di cui è il proprietario (praticamente, tutto ciò che sta sotto la sua home)  
È necessario un superuser per poter fare operazioni sulle altre cartelle

# USO DI CP, MV, RM CON PATTERN (I)

cp, mv, rm possono tutti essere usati in combinazione con pattern di file o cartelle per operare su più file contemporaneamente

```
marco@marco-VirtualBox:~/Pictures$ cd ~/Pictures
marco@marco-VirtualBox:~/Pictures$ ls
immagine-profilo.jpg
my_party_pictures
'Screenshot from 2021-09-03 19-02-32.png'
'Screenshot from 2021-09-03 19-16-55.png'
'Screenshot from 2021-09-06 13-23-16.png'
'Screenshot from 2021-09-06 15-09-51.png'
```

Qui sopra vediamo la composizione della cartella Pictures.

Vogliamo spostare tutti i file del tipo «Screenshot...» in una nuova cartella che chiameremo screenshot (che creiamo ora)

# USO DI CP, MV, RM CON PATTERN (II)

Anziché muovere i file a mano, posso notare che

1. Tutti i file iniziano con «Screenshot from »
2. Tutti i file hanno formato .png

Posso quindi identificare tutti questi file con un pattern del tipo

Screenshot\ from\ \*.png

```
marco@marco-VirtualBox:~/Pictures$ mkdir screenshot
marco@marco-VirtualBox:~/Pictures$ mv Screenshot\ from\ *.png ./screenshot
marco@marco-VirtualBox:~/Pictures$ ls
immagine-profilo.jpg  my_party_pictures  screenshot
marco@marco-VirtualBox:~/Pictures$ ls screenshot/
'Screenshot from 2021-09-03 19-02-32.png'
'Screenshot from 2021-09-03 19-16-55.png'
'Screenshot from 2021-09-06 13-23-16.png'
'Screenshot from 2021-09-06 15-09-51.png'
```

e scrivere mv Screenshot\ from\ \*.png ./screenshot

# USO DI CP, MV, RM CON PATTERN

Possiamo usare i pattern anche con cp e rm.

Proviamo, come esercizio, ad eseguire i seguenti compiti:

1. Posizioniamoci in screenshot
2. Copiamo i file ivi contenuti all'interno della cartella Pictures
3. Torniamo in Pictures
4. Cancelliamo tutti i file degli screenshot

Si poteva anche scrivere  
cp \* ..

Possibile

soluzione

```
marco@marco-VirtualBox:~/Pictures$ cd screenshot/
marco@marco-VirtualBox:~/Pictures/screenshot$ cp Screenshot\ from\ *.png ..
marco@marco-VirtualBox:~/Pictures/screenshot$ cd ..
marco@marco-VirtualBox:~/Pictures$ ls
immagine-profilo.jpg
my_party_pictures
screenshot
'Screenshot from 2021-09-03 19-02-32.png'
'Screenshot from 2021-09-03 19-16-55.png'
'Screenshot from 2021-09-06 13-23-16.png'
'Screenshot from 2021-09-06 15-09-51.png'
marco@marco-VirtualBox:~/Pictures$ rm Screenshot\ from\ *.png
marco@marco-VirtualBox:~/Pictures$ ls
immagine-profilo.jpg my_party_pictures screenshot
```

# ESERCIZIO INSIEME

## 1. Duplicazione di file:

- Vai nella cartella Documenti e copia il file `appunti.txt` nella directory principale ProgettoTerminale.

## 2. Spostamento e rinominazione:

- Sposta nella cartella principale ProgettoTerminale e rinomina il file `appunti.txt` appena copiato in `note.txt`.
- Sposta il file `bozza.txt` dalla cartella Documenti alla cartella principale ProgettoTerminale e rinominalo in `bozza_finale.txt`.

## 3. Eliminazione di file:

- Elimina il file `compiti.txt` dalla cartella Documenti.
- Vai nella cartella Immagini e elimina il file `foto2.png`.

## 4. Duplicazione con wildcard:

- Copia tutti i file `.txt` dalla cartella Documenti alla cartella principale ProgettoTerminale usando un carattere speciale.

# ALTRI COMANDI BASH – MAN

Se non sappiamo che cosa faccia di preciso un comando o che significato hanno i vari flag o parametri, abbiamo due opzioni

1. Cerchiamo in internet
2. Usiamo man <comando>
3. Per uscire dal manuale, premere q

Esempio: man ls

Schermata di testo  
«scorribile»

```
LS(1)                               User Commands                         LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for short options too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
        with -l, print the author of each file

    -b, --escape
        print C-style escapes for nongraphic characters

Manual page ls(1) line 1 (press h for help or q to quit)
```

Navigazione con tasti freccia, q per

# ALTRI COMANDI BASH – ECHO

Il comando `echo` viene utilizzato per stampare un messaggio o una stringa sul terminale.

## Sintassi di base:

- `echo [opzioni] [stringa]`

## Esempi pratici:

### 1. Stampare una stringa semplice:

- `echo "Ciao mondo!"`
- Risultato: `Ciao mondo!`

### 2. Aggiungere contenuto a un file:

- `echo "Questo è un test" >> file.txt`
- Questo comando aggiunge la stringa al file `file.txt` (crea il file se non esiste).

## Sequenze di escape comuni:

- `\n` : Nuova linea
- `\t` : Tabulazione
- `\\\` : Backslash

# SCRIVERE SU .TXT CON VIM

**Cos'è Vim:** Un editor di testo, utilizzato principalmente in ambienti Unix/Linux.

**Perché usare Vim:** Velocità, personalizzazione, disponibile su quasi tutti i sistemi, utile per scripting e programmazione.

Modalità di Vim

- **Modalità Normale:** La modalità predefinita per muoversi nel testo.
- **Modalità Inserimento:** Per inserire testo, attivabile con `i`.
- **Modalità Comando:** Per eseguire comandi, attivabile con `:`.
- **Modalità Visuale:** Per selezionare porzioni di testo.

# SCRIVERE SU .TXT CON VIM

## Apertura e Navigazione di Base

- **Aprire un file:** vim nomefile
- **Muoversi nel file:**
  - Frecce direzionali o:
  - h (sinistra), j (giù), k (su), l (destra)
  - gg (vai all'inizio), G (vai alla fine)

## Inserire e Modificare Testo

- **Modalità Inserimento:**
  - i: Inserire prima del cursore
  - a: Inserire dopo il cursore
  - o: Inserire una nuova riga sotto
- **Eliminare Testo:**
  - x: Cancella il carattere sotto il cursore
  - dw: Cancella una parola
  - dd: Cancella una riga intera

## Salvare e Uscire

- **Salvare i Cambiamenti:** :w
- **Uscire da Vim:** :q
- **Comandi combinati:**
  - :wq: Salva e chiudi
  - :q!: Chiudi senza salvare

## Copia, Taglia e Incolla

- **Copiare (Yank):**
  - yy: Copia la riga corrente
  - yw: Copia una parola
- **Tagliare (Delete):**
  - dd: Taglia una riga
  - dw: Taglia una parola
- **Incollare:** p

# ALTRI COMANDI BASH – CAT

cat ci permette di visualizzare il contenuto di un file (è una sorta di visualizzatore di testo super basilare).

Nella home directory personale ci sono due file, cestino.txt e cantol.txt, proviamo a visualizzarne il contenuto usando cat

```
marco@marco-VirtualBox:~$ cat cestino.txt
Il cestino (in inglese Trash o Recycle Bin) è una funzionalità dei sistemi operativi dotati di interfaccia grafica. È una componente specifica del file manager ed è spesso integrato nel desktop environment.

In genere è implementato come una o più directory non direttamente accessibili dall'utente. La sua funzione è quella di contenere i file cancellati dal PC e permettere di navigare tra essi, annullarne l'eliminazione e lo spostamento nel cestino (riportandoli alla loro posizione originaria) oppure eliminarli definitivamente dall'hard disk o dal supporto fisico in cui erano originariamente presenti, recuperando memoria.
```

# ALTRI COMANDI BASH – LESS

Il problema con cat è che stampa direttamente tutto il contenuto del file nella shell.

Se il file è molto grande (come nel caso di cantol.txt) mi tocca scorrere all'infinito verso l'alto per recuperare l'inizio del file.

Senza contare che la memoria del terminale è limitata, e quindi viene conservata una storia limitata di comandi e output.

Un'opzione più fruibile è less, che mi permette di scorrere il file lungo il terminale, come accadeva nel caso di man

less cantol.txt

Per uscire premere  
q

Nel mezzo del cammin di nostra vita  
mi ritrovai per una selva oscura,  
ché la diritta via era smarrita.

Ahi quanto a dir qual era è cosa dura  
esta selva selvaggia e aspra e forte  
che nel pensier rinova la paura!

Tant' è amara che poco è più morte;  
ma per trattar del ben ch'i vi trovai,  
dirò de l'altre cose ch'i v'ho scorte.

Io non so ben ridir com' i' v'intrai,  
tant' era pien di sonno a quel punto  
che la verace via abbandonai.

Ma poi ch'i fui al piè d'un colle giunto,  
là dove terminava quella valle  
che m'avea di paura il cor compunto,

guardai in alto e vidi le sue spalle  
vestite già de' raggi del pianeta  
che mena dritto altri per ogn'e calle.

Allor fu la paura un poco queta,  
che nel lago del cor m'era durata  
:

# ALTRI COMANDI BASH – HEAD

Un'alternativa a less, soprattutto nel caso in cui mi interessasse solamente dare un'occhiata alle prime righe del file, è head

Con head leggo solo le prime 10 righe del file

```
marco@marco-VirtualBox:~$ head cantoI.txt

Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.

Ahi quanto a dir qual era è cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinova la paura!
```

# ESERCIZIO INSIEME

## **Visualizzazione del contenuto di un file:**

- Vai nella cartella principale ProgettoTerminale.
- Visualizza il contenuto di file note.txt

## **Aggiunta di testo a un file:**

- Aggiungere la frase "Questi sono i miei appunti" alla fine del file note.txt.
- Visualizza il contenuto di file note.txt

## **Modifica del file:**

- Aprire il file note.txt.
- Modificare la frase “Questi sono i miei appunti” in “Questi sono i miei pazzeschi appunti”
- Salvare le modifiche ed uscire

## **Guida ai comandi con man:**

- Usa il comando `man` per visualizzare il manuale di `cp` e prendi nota di un'opzione che può essere utilizzata per mantenere i permessi originali del file durante la copia.
- Consulta il manuale di `rm` e cerca l'opzione per confermare l'eliminazione di ciascun file prima di procedere.

# ALTRI COMANDI BASH – CLEAR

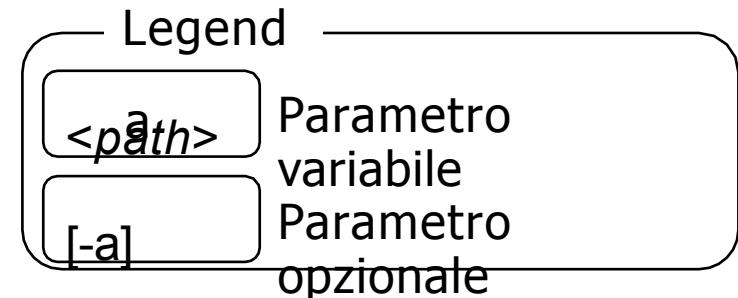
Dal momento in cui abbiamo un terminale pieno di comandi pregressi, possiamo usare clear per cancellare tutto e ripartire con una schermata «fresca».

Proviamo ad usarlo sulla nostra shell e a vedere cosa succede.

Che differenza c'è fra usare clear e chiudere e aprire una nuova schermata del terminale?

# RIASSUNTO DEI COMANDI FILESYSTEM

cd	<i>Change Directory</i>	cd <path>
ls	<i>List files</i>	ls [-a] [<path>]
pwd	<i>Print Working Dir</i>	pwd
touch	<i>Create file</i>	touch <path(s)>
mkdir	<i>Make directory</i>	mkdir <path(s)>
cp	<i>Copy</i>	cp [-r] origin(s) destination
mv	<i>Move</i>	mv origin(s) destination
rm (rmdir)	<i>Remove (directory)</i>	rm [-r] <path(s)> rmdir <path(s)>



«Navigazione  
filesystem»

«Manipolazione  
filesystem»

NB: in origin(s) è possibile  
anche indicare pattern

# RIASSUNTO DEI COMANDI

## ALTRI COMANDI

		Legend
		<path> Parametro variabile
man	Command manual	man <command>
cat	Output file content	cat <file>
head	Head of file content	head <file>
less	Scroll file content	less <file>
clear	Clear screen	clear
xdg-open	Open file	xdg-open <file>

Visualizzazione contenuto file

# ALTRI COMANDI BASH – VISUALIZZAZIONE PERMESSI

- **Comando:** `ls -l`
- Permette di vedere i permessi dei file nella directory corrente.
  - I permessi sono organizzati in tre categorie:
    - **u:** proprietario (user)
    - **g:** gruppo (group)
    - **o:** altri (others)
- Formato output:
  - **r** = read (lettura)
  - **w** = write (scrittura)
  - **x** = execute (esecuzione)
- I primi 3 caratteri si riferiscono al **proprietario**.
- I successivi 3 caratteri si riferiscono al **gruppo**.
- Gli ultimi 3 caratteri si riferiscono agli **altri** utenti.
- Esempio: `-rwxr-xr--`

# ALTRI COMANDI BASH – MODIFICA PERMESSI

- **Comando:** `chmod`
- **Sintassi:** `chmod [permessi] [file]`
- Modifica i permessi di un file o una directory.
- Esempio:
  - Aggiungi permesso di esecuzione al proprietario: `chmod u+x file.txt`
  - Rimuovi permesso di scrittura per gli altri: `chmod o-w file.txt`

# ESERCIZIO INSIEME

## **Visualizzazione dei dettagli dei file:**

- Usa il comando `ls -l` per visualizzare i dettagli (inclusi i permessi) di tutti i file nella cartella `ProgettoTerminale` e nelle sue sottocartelle.

## **Modifica dei permessi su un file:**

- Modifica i permessi del file `note.txt` in modo che solo il proprietario possa leggerlo e scriverlo, mentre gli altri utenti non possano né leggerlo né modificarlo.

## **Verifica delle modifiche:**

- Usa di nuovo il comando `ls -l` per verificare che i permessi siano stati modificati correttamente per i file e le directory.

# ALTRI COMANDI BASH – VISUALIZZAZIONE PROCESSI

- **Comando: ps**
  - Mostra una lista dei processi attivi dell'utente corrente riguardo la sessione del terminale considerata
    - `ps aux` per vedere tutti i processi indipendentemente dall'utente o dalla sessione
    - `Ps -ef` mostra i dettagli del processo
- **Comando: top**
  - Mostra i processi attivi in tempo reale, con l'utilizzo di CPU e memoria.

# ALTRI COMANDI BASH – CARATTERISTICHE DEL PC

## Visualizzare le specifiche del PC

- Usa `lscpu` per vedere le informazioni sulla CPU (Mac: `sysctl -a | grep machdep.cpu`)
- Usa `free -h` per visualizzare l'uso della RAM.
- Usa `df -h` per vedere lo spazio su disco disponibile.
- Usa `uname -a` per visualizzare le informazioni generali del sistema.
- Usa `cat /etc/os-release` per scoprire la versione del sistema operativo (`sw_vers`).

# ESERCIZIO RIASSUNTIVO

1. **Navigazione e creazione di directory:**

- Verifica in quale directory ti trovi attualmente.
- Crea una nuova directory chiamata `progetto` all'interno della tua directory corrente.
- Sposta nella nuova directory appena creata.

2. **Creazione e gestione di file:**

- Crea un nuovo file vuoto chiamato `README.md` nella directory `progetto`.
- Crea altri due file vuoti chiamati `main.py` e `config.json` utilizzando un comando singolo.

3. **Visualizza il contenuto della directory:**

- Mostra il contenuto della directory con i dettagli completi (come permessi, dimensioni, ecc.).

4. **Modifica dei file:**

- Apri il file `README.md` con Vim e scrivi una breve descrizione del progetto.

5. **Copia e modifica dei file:**

- Crea una copia del file `main.py` e rinominalo in `main_backup.py`.

6. **Permessi e gestione:**

- Cambia i permessi del file `main.py` per rendere eseguibile solo per il proprietario.

7. **Pulizia e visualizzazione:**

- Visualizza il contenuto del file `README.md` sul terminale.
- Cancella il file `config.json`.

8. **Utilità e pulizia finale:**

- Cancella la schermata del terminale per rendere la visualizzazione pulita.
- Utilizza il comando per visualizzare la documentazione del comando `chmod`.

# 4. LINUX 101

Installazione applicazioni in linux

# INSTALLAZIONE APPLICAZIONI IN UBUNTU

A differenza di Windows, dove le applicazioni vengono scaricate dai siti dei produttori ed eseguiti tramite file .exe, in Ubuntu si preferisce affidarsi a cosiddetti ***package manager***

Installazione applicazioni

Aggiornamenti

Gestione dipendenze

# APT E UBUNTU SOFTWARE

- **Cos'è apt?**

- apt (Advanced Package Tool) è un gestore di pacchetti utilizzato in sistemi basati su Debian (come Ubuntu).
- Per software dal terminale permette di:
  - installare,
  - aggiornare
  - rimuovere

- **Perché usare apt?**

- Semplice gestione dei pacchetti.
- Accesso a un vasto repository di software.
- Mantiene il sistema aggiornato e sicuro.

## Comandi principali di apt

### 1. Aggiornare la lista dei pacchetti

- Comando: `sudo apt update`
- **Descrizione:** Aggiorna l'elenco dei pacchetti disponibili nel repository.

### 2. Aggiornare i pacchetti installati

- Comando: `sudo apt upgrade`
- **Descrizione:** Aggiorna tutti i pacchetti installati alle versioni più recenti disponibili.

### 3. Installare un pacchetto

- Comando: `sudo apt install <nome_pacchetto>`
- **Descrizione:** Installa un pacchetto specifico dal repository.

### 4. Rimuovere un pacchetto

- Comando: `sudo apt remove <nome_pacchetto>`
- **Descrizione:** Rimuove un pacchetto dal sistema.

## Altri comandi utili

### 1. Pulizia dei pacchetti non necessari

- Comando: `sudo apt autoremove`
- **Descrizione:** Rimuove i pacchetti inutilizzati che sono stati installati automaticamente come dipendenze.

### 2. Rimuovere i pacchetti scaricati

- Comando: `sudo apt clean`
- **Descrizione:** Rimuove i file dei pacchetti scaricati che non sono più necessari (libera spazio su disco).

### 3. Verificare i dettagli di un pacchetto

- Comando: `apt show <nome_pacchetto>`
- **Descrizione:** Mostra informazioni dettagliate su un pacchetto.

# APT E UBUNTU SOFTWARE

Slide: Installazione e Utilizzo di `curl` con `apt`

## 1. Installare `curl`

- **Comando:**  
`sudo apt update`
- `sudo apt install curl`
  - `curl` è un potente strumento da riga di comando utilizzato per trasferire dati da o verso un server, comunemente utilizzato per testare API e scaricare file.

## 2. Perché `curl` è utile?

- **Testare rapidamente una connessione a un server:**  
Verifica la risposta di un sito web o un'API senza bisogno di un browser o di un'interfaccia grafica.
- **Scaricare file da internet:**  
Puoi usare `curl` per automatizzare il download di file direttamente nel terminale.

## 3. Esempio di utilizzo di `curl`

- **Comando**  
`curl https://raw.githubusercontent.com/gpietroP/PreCorso-INF-unITS/main/scaricami.txt`
  - Questo comando scarica e mostra il contenuto della pagina web di "<https://example.com>" nel terminale.

## 4. Esempio pratico: Scaricare un file

- **Comando:**  
`curl -O https://raw.githubusercontent.com/gpietroP/PreCorso-INF-unITS/main/scaricami.txt`
  - **Descrizione:** Scarica il file "file.txt" dalla pagina web "<https://example.com>" e lo salva nella directory corrente.

# COMANDI BASE PER APT (I)

Installazione di un'applicazione: apt install <NomeApp>

```
marco@marco-VirtualBox:~$ apt install emacs
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
```

Per poter installare, dobbiamo essere superuser

Ci sono due modi per diventare superuser

1. Loggarsi come superuser (nome utente: root) sconsigliato
2. Eseguire un singolo comando come superuser (*keyword sudo*)



# COMANDI BASE PER APT (II)

Installazione

applicazione:

`sudo apt install <NomeApp>`

Prima di installare un'applicazione, è buona norma effettuare un **aggiornamento del catalogo di APT**, così esso può sapere se vi sono versioni più recenti del programma da installare e di tutte le sue dipendenze.

`sudo apt update`



Update non aggiorna alcun programma installato, ma effettua solamente un *refresh* del catalogo

Disinstallazione

applicazione:

`sudo apt remove <NomeApp>`

Aggiornamento applicazioni installate:  
`sudo apt upgrade`

# INSTALLAZIONE TRASH-CLI

trash-cli è un'applicazione per il terminale  
(NB: *CLI* = *Command Line Interface*) che ci  
consente di **spostare un file o cartella in  
cestino**, senza rimuoverlo definitivamente.

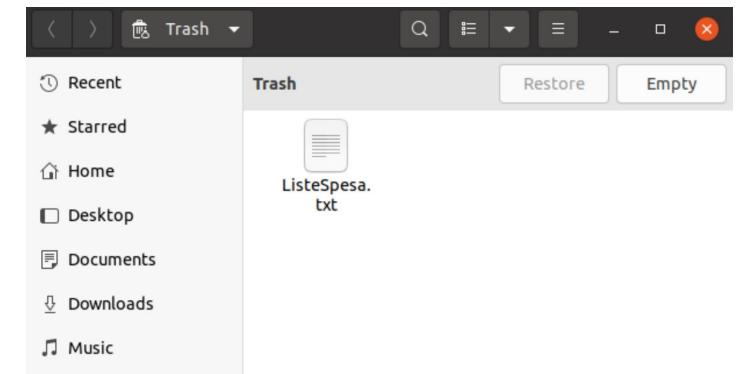
*Che comando utilizziamo per  
l'installazione?*

sudo apt update

sudo apt install trash-cli

Ora possiamo  
rimuovere i file in  
maniera più sicura  
utilizzando il comando  
`trash-put` anziché `rm`.

```
marco@marco-VirtualBox:~/Documents/Documenti_casa$ trash-put ListeSpesa.txt
marco@marco-VirtualBox:~/Documents/Documenti_casa$ ls ~/.local/share/Trash/files/
ListeSpesa.txt
marco@marco-VirtualBox:~/Documents/Documenti_casa$
```



# RIASSUMENDO

Ubuntu utilizza un package manager a linea di comando chiamato APT

Tramite APT possiamo installare e aggiornare le applicazioni

Per usare APT abbiamo bisogno dei privilegi di amministratore

`sudo apt update`

Aggiorna  
catalogo

`sudo apt install <NomeApp>`

`sudo apt remove <NomeApp>`

`sudo apt upgrade`

Aggiorna applicazioni  
installate

# LA PROSSIMA VOLTA

Riprenderemo un attimo il concetto di Markdown per introdurre l'evoluzione di un file nel tempo

Passeremo quindi a motivare la necessità del *versioning* e della gestione della storia di un file o di un insieme di file

Vedremo uno dei possibili programmi di Version Control System, git, che verrà usato estensivamente nei corsi di tutti i livelli, sia della triennale che della magistrale

# PREREQUISITI PER LA PROSSIMA LEZIONE

## **Installare Visual Studio Code**

- Su Ubuntu...
  1. Installare snap (se non già presente) `sudo apt install snap`
  2. Installare VSCode con snap: `sudo snap install --classic code`
- Su Mac...
  1. Da terminale: `brew install --cask visual-studio-code`

# PREREQUISITI PER LA PROSSIMA LEZIONE

## Installare git

- Su Windows...
  1. Download the installer ([Git for Windows](#)) from the VScode website
  2. Runnare l'installer con le impostazioni di default
  3. Aprire Git Bash e verificare la corretta installazione con —git version
- Su Ubuntu...
  1. Dal terminale: sudo apt update
  2. Dal terminale: sudo apt install git
- Su Mac...
  1. Probabilmente già presente se avete installato homebrew (controllare con git —version)
  2. Altrimenti: brew install git

# PREREQUISITI PER LA PROSSIMA LEZIONE

- Una volta che avete installato git dovete anche configurarlo con il vostro account che avete creato per GitHub

bash

 Copia codice

```
git config --global user.name "Your Name"  
git config --global user.email "your.email@example.com"
```

# PREREQUISITI PER LA PROSSIMA LEZIONE

- Verifichiamo di aver configurato correttamente git!

Run the following command to display your Git configuration settings:

```
bash
```

 Copia codice

```
git config --global --list
```

This will show key details such as your username and email. Look for the lines with:

```
css
```

 Copia codice

```
user.name=Your Name  
user.email=your.email@example.com
```

If these match the account you want to use, Git is correctly configured.