

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
Sottosistem
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.

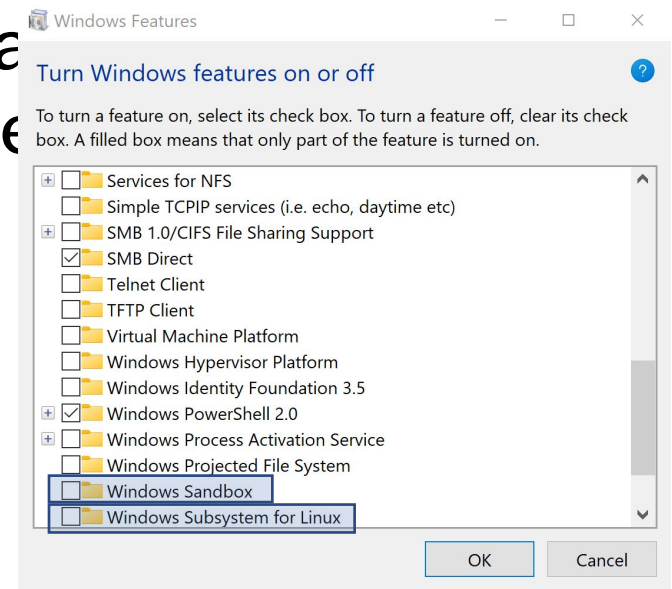
Vogliamo fare in modo di *emulare* il processo secondo cui installiamo il sistema operativo inserendo il DVD nel lettore, senza però avere bisogno di un DVD fisico!



POSSIBILI FONTI DI INCOMPATIBILITÀ

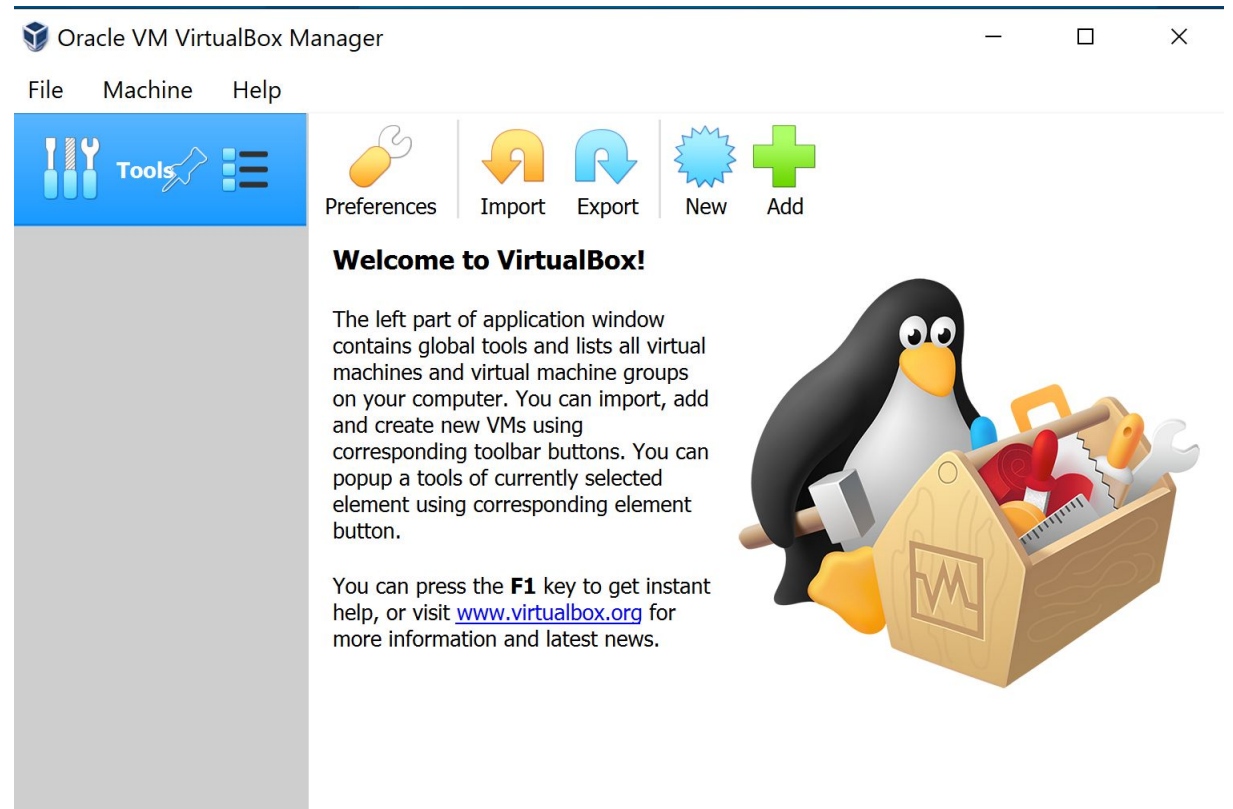
VirtualBox non funziona se su Win10 è già attivo il Linux Subsystem e se sono attive altre funzionalità come HyperV e Windows Sandbox.

Per disattivare questi servizi, attivare la finestra premendo la combinazione di tasti + R e digitare Cercare i 3 nomi sopra indicati e togliere il tick. Riavviare il computer.



COSA ACCADE ORA?

Apriamo VirtualBox e
selezioniamo 
Segue lezione pratica



4. LINUX 101

Interfaccia *desktop*

L'INTERFACCIA DESKTOP DI UBUNTU

File explorer di Ubuntu (si chiama *Nautilus*)

Barra applicazioni (analoga a Windows 10)

Start menu (analogo a Windows 10)



Cestino

Interfaccia

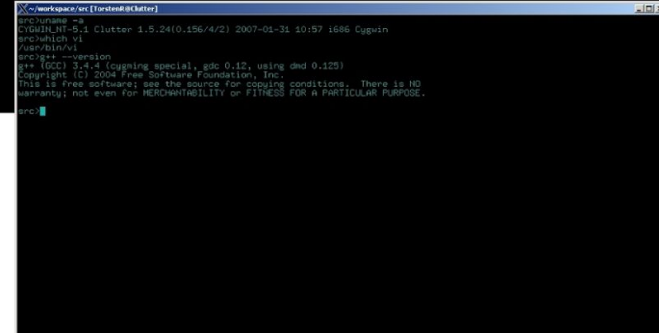
4. LINUX 101

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.

A terminal window titled "workspace [larsen@clater]" showing the output of the 'uname -a' command. The output lists system details including kernel version (2.6.18-9.el1), architecture (i386), and compiler information (gcc 4.1.2).

```
workspace [larsen@clater]
larsen@clater ~$ uname -a
Linux clater 2.6.18-9.el1 i386 GNU/Linux
larsen@clater ~$ cat /etc/redhat-release
Red Hat Enterprise Linux (EL5) 5.4
larsen@clater ~$ gcc --version
gcc (GCC) 4.1.2 (Gentoo 4.1.2/0.0)
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
larsen@clater ~$
```

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

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

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

Output dell'OS: composizione della cartella indicata

```
marco@marco-VirtualBox: ~/Downloads/Telegram Desktop
marco@marco-VirtualBox:~$ cd Documents/
marco@marco-VirtualBox:~/Documents$ ls
marco@marco-VirtualBox:~/Documents$ cd ..
marco@marco-VirtualBox:~$ cd Pictures/
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_Missione.pdf
marco@marco-VirtualBox:~/Downloads/Telegram Desktop$
```

Cartella corrente = *working directory* → se eseguo comandi per filesystem, li esegue in questa cartella

Riga corrente → attesa comando utente

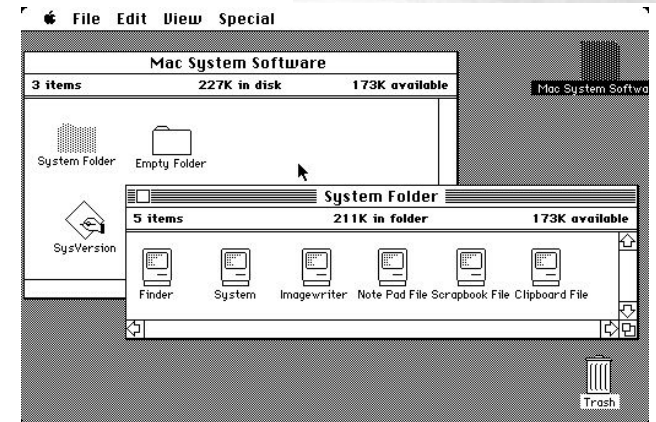
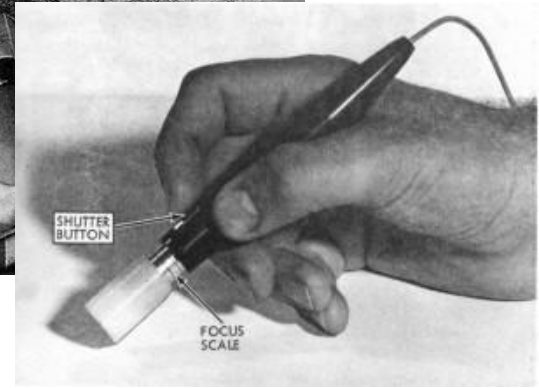
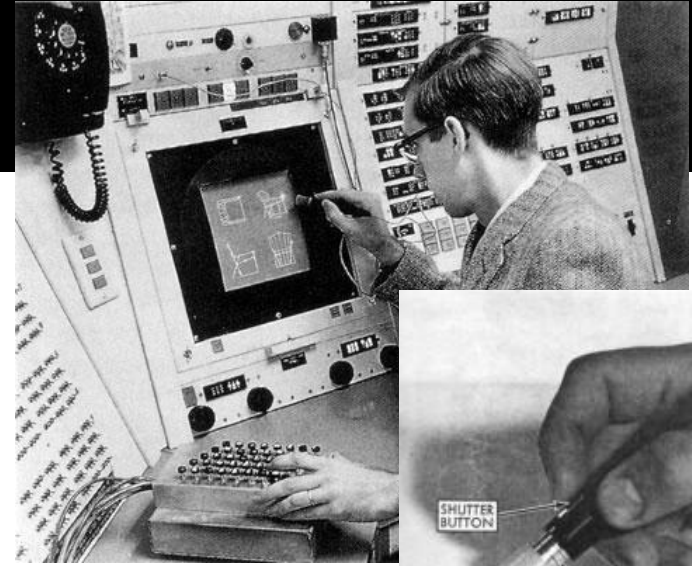
BREVE STORIA DELL'INTERAZIONE UOMO-MACCHINA

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.

I NOSTRI 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)

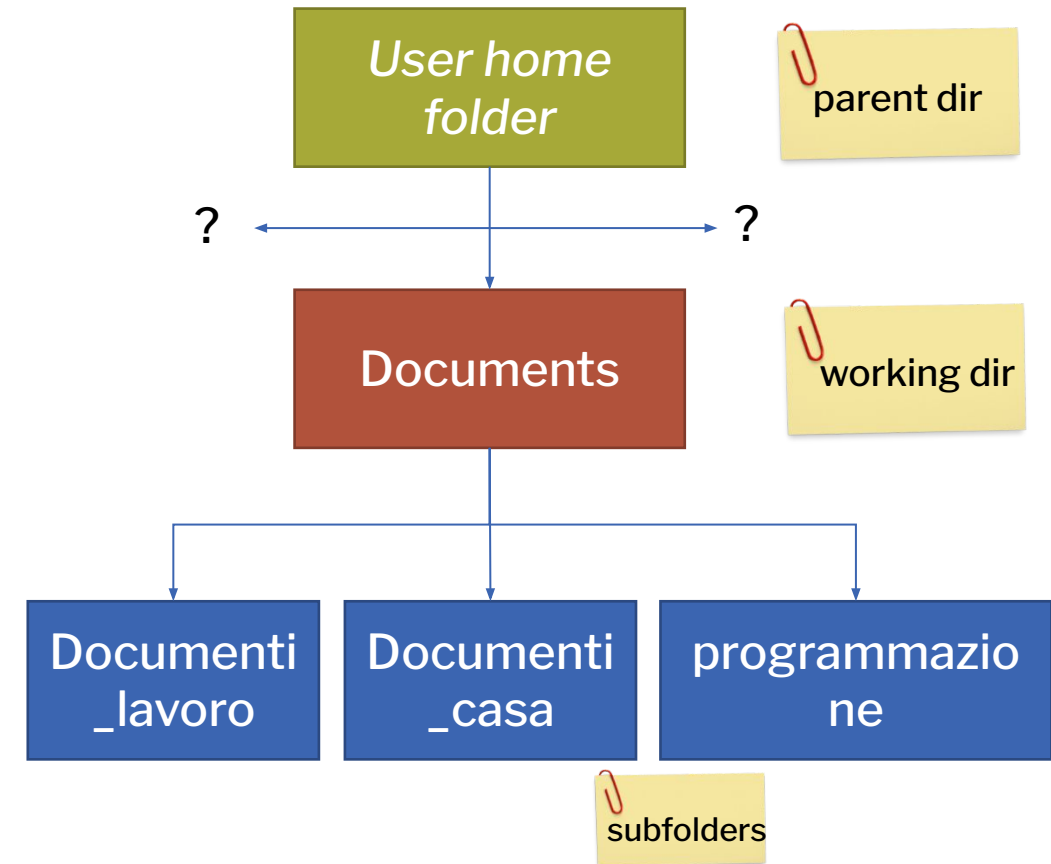
P. assoluta
Posizione della *working directory* all'interno del sistema *ad albero* del filesystem

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

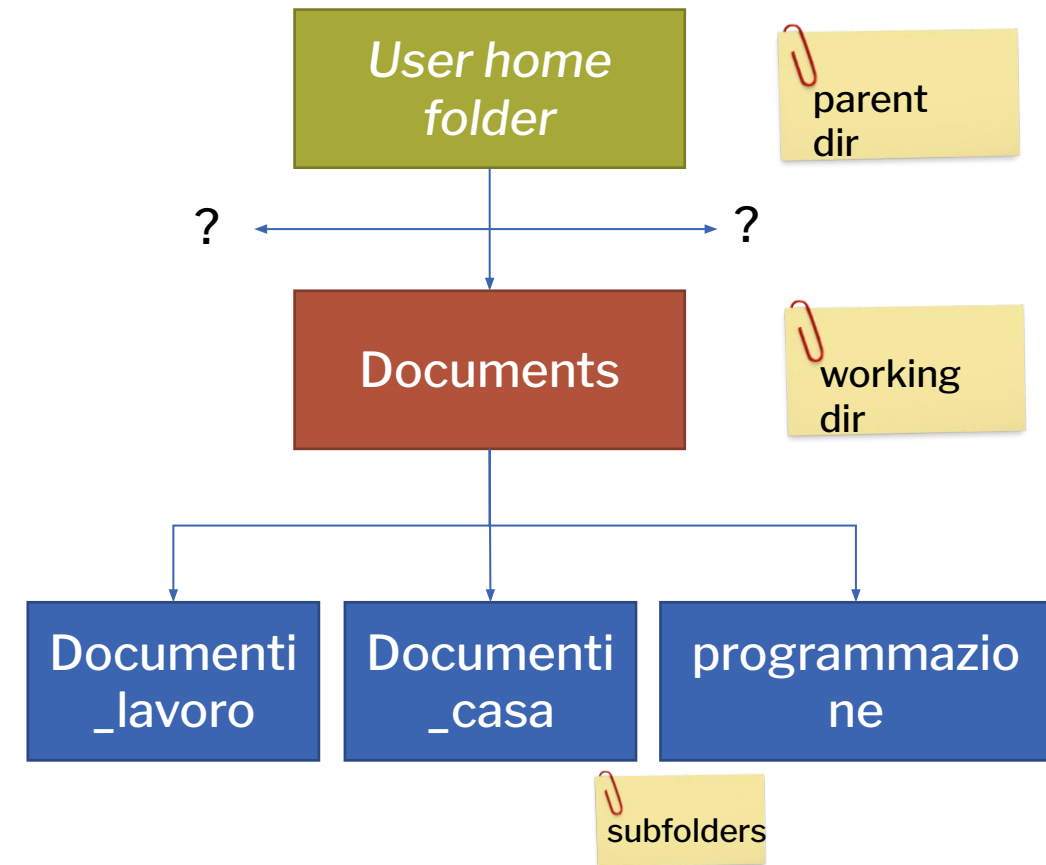
Proviamo a muoverci in una delle sottocartelle - programmazione

cd programmazione

Vogliamo tornare indietro in - Documents

cd Documents

Che cosa succede?



CD – PARENT DIRECTORY

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

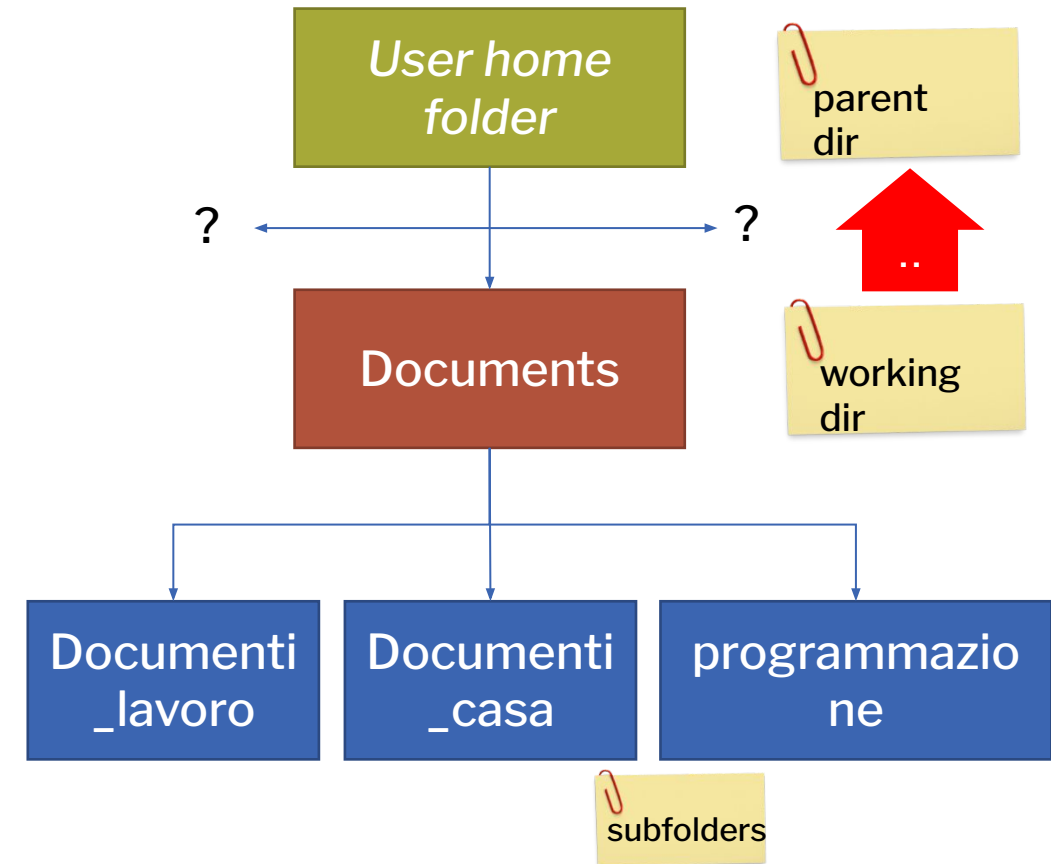
→ riceviamo un errore

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

In Bash alcuni caratteri hanno significati precisi

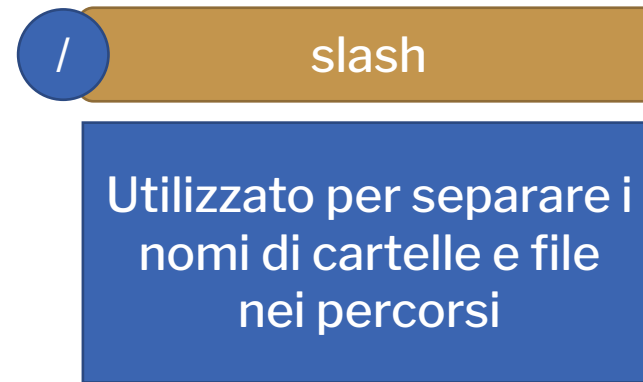
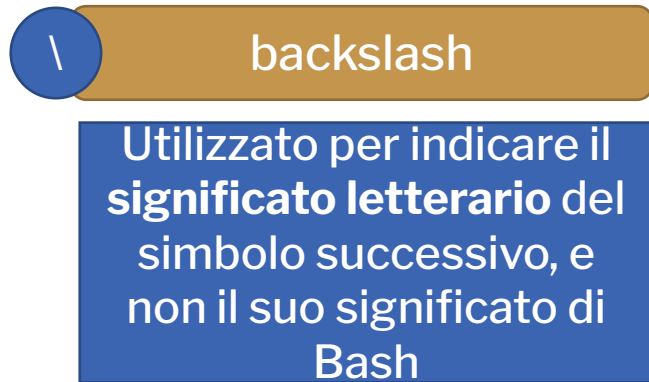
- Abbiamo già visto che `..` viene usato per tornare alle parent dir
- E che lo **spazio** sia utilizzato come separatore di stringhe/comandi

.	Cartella corrente
..	Cartella parent
~	User home folder
?	Wildcard per carattere singolo
*	Wildcard per sequenze caratteri

vi sono altri caratteri che imparerete a conoscere in corsi più avanzati

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*.



```
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**

equivalente a

ls

Lista file cartella corrente



ls .

ls percorso/cartella

Lista file cartella indicata

```
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

Vi è ancora un tipo di parametro che richiede un **valore aggiuntivo**

ls Documents **--hide=<pattern>**

ESEMPIO DI PARAMETRI OPZIONALI CON LS

Mostra tutti i file di una cartella

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

sono presenti anche i percorsi relativi . e ..

Mostra anche i file nascosti!
In Unix file e cartelle nascosti si identificano dal punto iniziale: .file_nascosto

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

Lista in ordine discendente di modifica

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

NB: i short flags possono essere concatenati

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

LS --HIDE

Possiamo anche utilizzare il parametro **--hide**, che richiede un valore aggiuntivo, il pattern del file da nascondere.

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

Possiamo passare il nome di un file o cartella per escluderlo dalla lista

```
marco@marco-VirtualBox:~/Documents$ ls programmazione --hide=hello*  
array.py
```

Se vogliamo indicare un pattern generico dobbiamo usare le wildcard viste prima

--hide=hello* → escludi tutti i file e cartelle contenente la stringa «hello» e qualsiasi numero di caratteri successivi

?

Wildcard per carattere singolo

*

Wildcard per sequenze caratteri

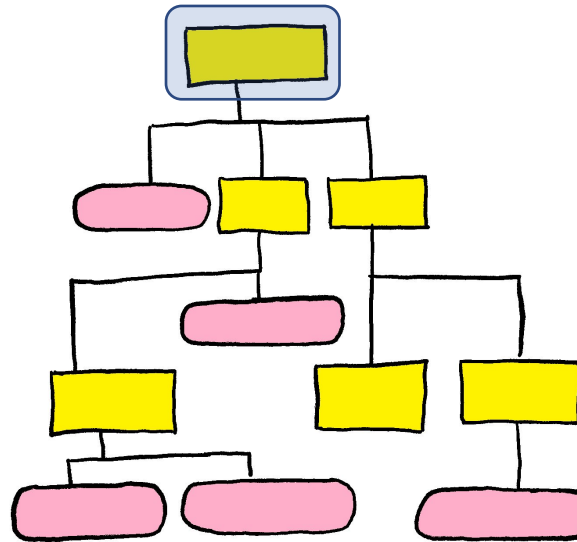
LA STRUTTURA DEL FILESYSTEM DI UBUNTU

Esercizio

Come facciamo a trovare la *cartella base (radice)* del filesystem Linux?

```
marco@marco-VirtualBox:/$ pwd
```

«/» è il percorso della radice del filesystem



Soluzione: apriamo il terminale e continuiamo ad eseguire `cd ..` (+ `pwd`) fino a quando non possiamo più scendere di livello

Esercizio

Quali sono (tutte) le cartelle e i file contenuti in questa cartella?

Soluzione:
`ls -a /`

```
marco@marco-VirtualBox:/$ ls -a
.   boot  etc    lib32  lost+found  opt   run   srv   tmp
..  cdrom  home  lib64  media       proc  sbin  swapfile  usr
bin  dev    lib   libx32  mnt         root  snap  sys    var
```

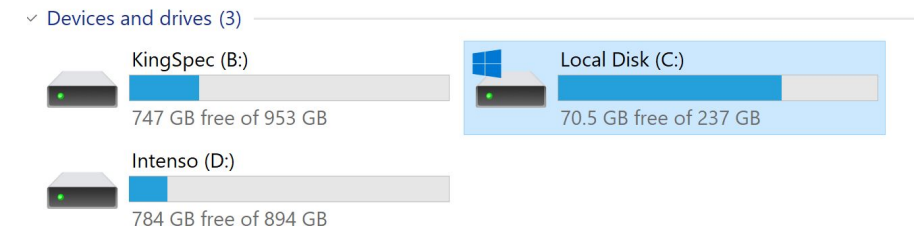
RELAZIONE CON IL FILESYSTEM DI WINDOWS

Il filesystem di Windows non presenta una radice univoca per tutto il sistema

Ogni disco fisso (o partizione di disco) utilizzato dal computer ha un nome composto da una lettera (es. C:/, D:/...)

Ogni path contiene, in testa, il nome del disco (es. C:/Windows)

Il disco stesso è la radice del filesystem per quanto riguarda quel disco



Il disco di *default* è quello dove è installato Windows (usualmente chiamato C:/ per convenzione)

FILESYSTEM UBUNTU – CARTELLE DI SISTEMA

```
marco@marco-VirtualBox:/$ ls -a
.   boot  etc    lib32  lost+found  opt   run   srv   tmp
..  cdrom  home  lib64  media       proc  sbin  swapfile  usr
bin  dev    lib   libx32  mnt         root  snap  sys      var
```

bin contiene i file eseguibili relativi ai comandi da terminale, come cd, ls...

boot contiene i file fondamentali per l'avvio del sistema

dev contiene i file relativi alle varie periferiche del sistema (dischi fissi compresi)

home contiene le cartelle home dei vari utenti del sistema

media permette di esaminare il contenuto delle periferiche rimovibili (es. chiavette USB)

root è la home del superuser root: vale a dire un super-utente che è *proprietario* dell'intero sistema

tmp contiene i file temporanei

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

Esercizio

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

Tramite «;» posso concatenare più comandi da eseguire in sequenza

```
cd precorsi_informatica; touch comandi_bash.txt
```

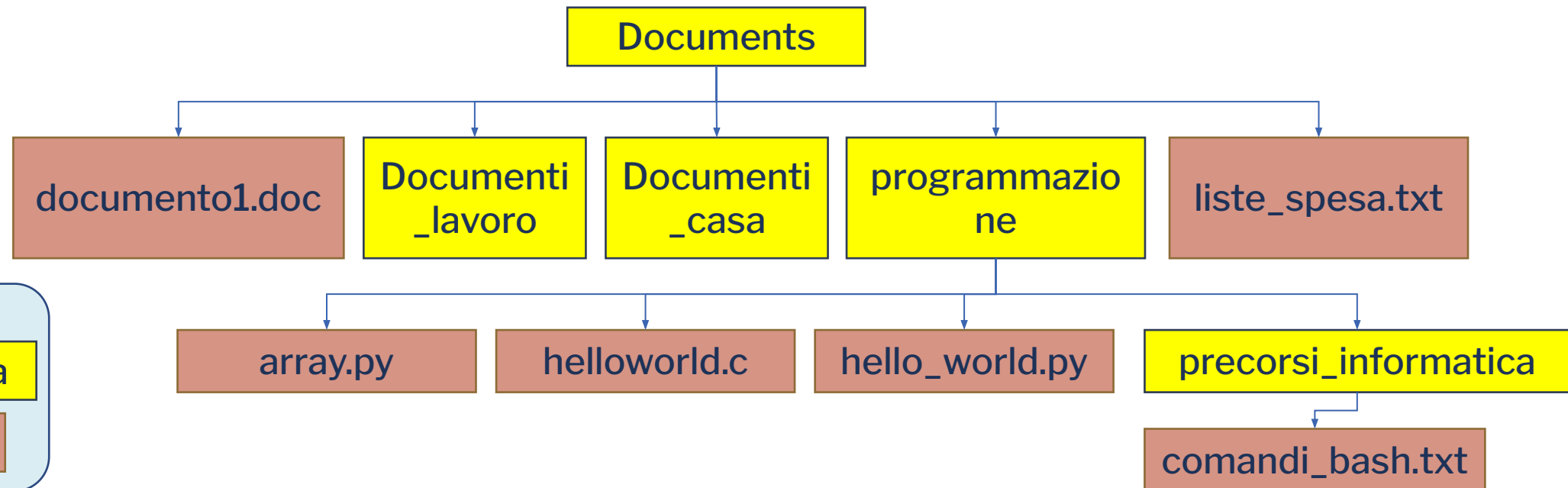
```
touch precorsi_informatica/comandi_bash.txt
```

```
touch ~/Documents/precorsi_informatica/comandi_bash.txt
```

ESERCIZIO – RIPRODUZIONE DELLA STRUTTURA DI DOCUMENTS

Esercizio

Ora avete tutti gli strumenti a disposizione per ricreare, sulla vostra macchina Linux, la struttura della cartella Documents



Legenda

cartella

file

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...



```
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/ pre  
corsi_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

Change Directory

cd <path>

ls

List files

ls [-a -l -t] [<path>]

pwd

Print Working Dir

pwd

touch

Create file

touch <path(s)>

mkdir

Make directory

mkdir <path(s)>

cp

Copy

cp [-r] origin destination

Legend

<path>

Parametro
variabile

[-a]

Parametro
opzionale

«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_copia/ 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 ListeSpesa.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!



rm comandi_bash_copia.txt

rm -r precorsi_informatica_copia

Anche
rmdir
se
vuota

```
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 con `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

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

```
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'
```

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
```

ESERCIZI

Andrea ha creato una cartella PrivatePictures all'interno di Pictures. Vuole rendere questa cartella privata. Come può fare?

Aiutino: in Unix, le cartelle private iniziano tutte col punto "."

Documents.

- 1) Deve trasferire solo i file presenti direttamente nella cartella Documents nella sua chiavetta USB
- 2) Deve trasferire tutti i file di Documents (sottocartelle comprese) nella sua chiavetta USB

Pierre ha per errore messo alcuni documenti nella cartella Pictures, e vorrebbe spostarli nella cartella Documents

- 1) i file con estensione .docx vanno nella cartella Documenti_lavoro
- 2) i file con estensione .doc, .xls, .xlsx vanno nella cartella Documenti_casa
- 3) i file con estensione .txt vanno invece rimossi

Tip: i comandi cp, rm, mv possono operare su file o pattern multipli, che vanno specificati prima della cartella di destinazione

Es: cp file1 file2 file3 cartella_dest

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>

Esempio: man ls

Schermata di testo

```
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 speci-
  fied.

  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 – 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 permette 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

```
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 altrui per ogni 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!
```

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?

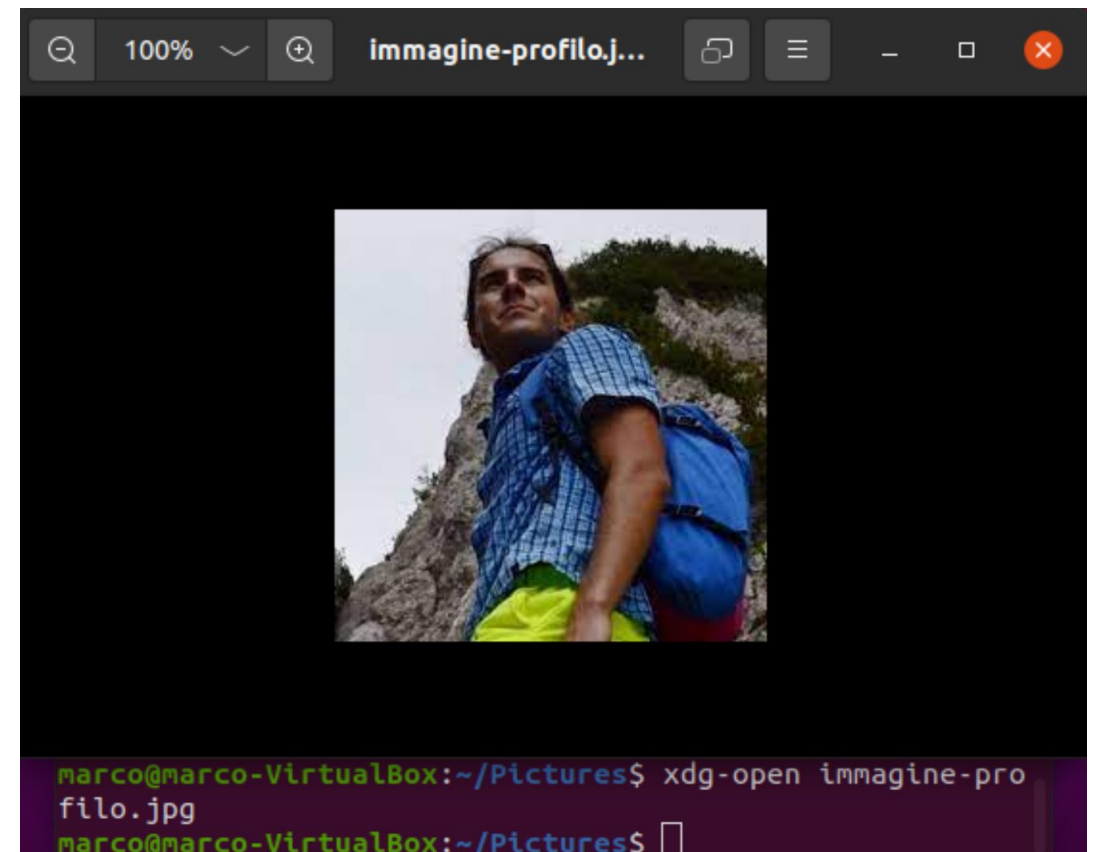
COMANDO SPECIFICO UBUNTU – XDG-OPEN

xdg-open è utilizzato per aprire un determinato file con la applicazione di default per il suo formato

Che cos'è un'applicazione di default?

Il sistema operativo tiene un *dizionario* delle applicazioni che sono deputate ad aprire, di default, un determinato tipo di file.

Nota: per aprire un file a partire dal terminale possiamo anche scrivere
nautilus .
e fare doppio click sul file desiderato



RIASSUNTO DEI COMANDI FILESYSTEM

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

Legend

- <path> Parametro variabile
- [-a] Parametro opzionale

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

RIASSUNTO DEI COMANDI

ALTRI COMANDI

man	Command manual	man <command>	
cat	Output file content	cat <file>	Visualizzazione contenuto 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>	

Legend

`<path>` Parametro variabile

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

Ubuntu fa affidamento ad un package manager

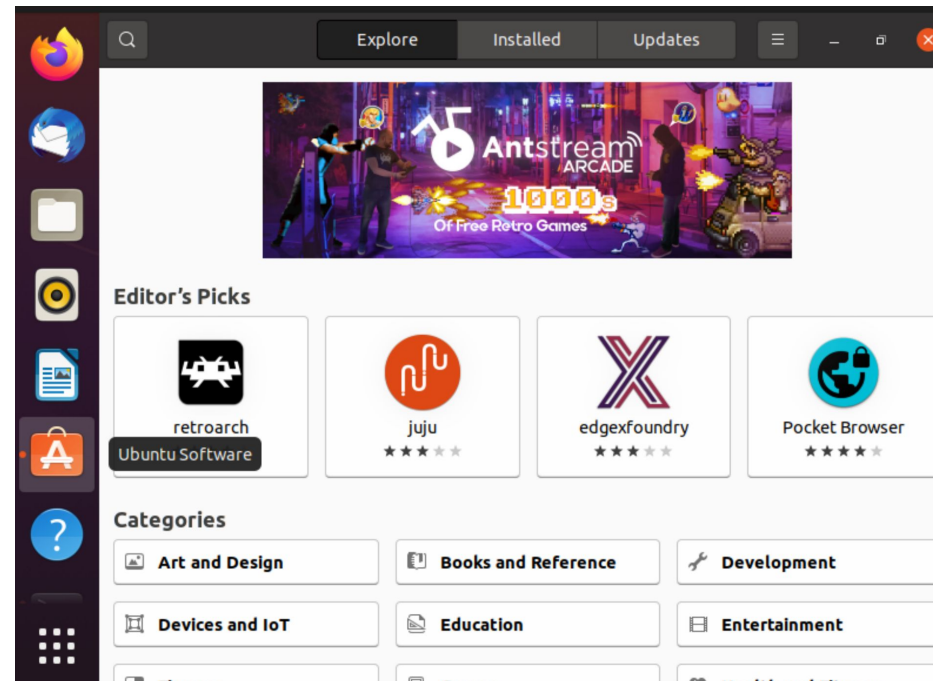
APT
Advanced Packaging Tool

Package manager a linea di comando

Le funzionalità vengono richiamate da terminale

Do principalmente comandi di installazione e aggiornamento

Esistono applicazioni più recenti (es. Ubuntu Software) con GUI



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-frontent - open (13: Permission
denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), are
you root?
```

Per poter
installare,
dobbiamo essere
superuser

Ci sono due modi per diventare superuser

1. Loggarsi come superuser (nome utente: root) sconsigliat
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



Aggiornamento applicazioni
installate:
`sudo apt upgrade`

Disinstallazione

applicazione:
`sudo apt remove <NomeApp>`

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.

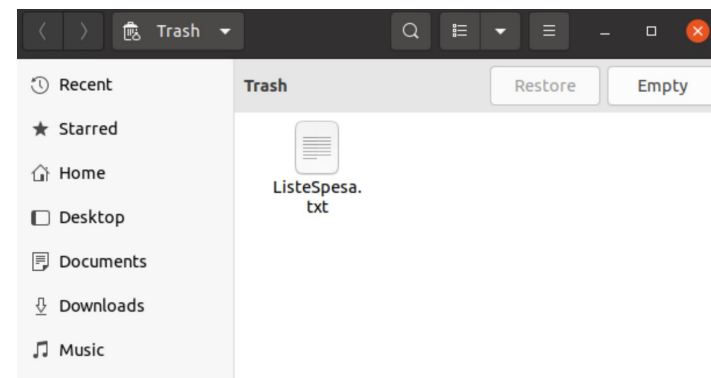
Che comando utilizziamo per l'installazione?

sudo apt update

sudo apt install trash-cli

Ora possiamo rimuovere i file in maniera più sicura utilizzando il comando

```
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
 - NB: Su Ubuntu...
 1. Installare snap (se non già presente) `sudo apt install snap`
 2. Installare VSCode con snap: `sudo snap install --classic code`
- Installare Git
 - Su Windows: installare [Git for Windows](#) seguendo la procedura indicata qui tinyurl.com/98fs8cmu
 - Su Ubuntu: `sudo apt install git`
 - Su Mac già presente se avete installato homebrew
- creare un account su github.com