

INFORMATICA

Intelligenza Artificiale & Data Analytics

Precorsi a.a. 2025/26

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!



INSTALLAZIONE COME MACCHINA VIRTUALE – FASI PRELIMINARI

1. Installare VirtualBox

- Vai al sito ufficiale: [virtualbox.org](https://www.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
 - **Windows:** 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).

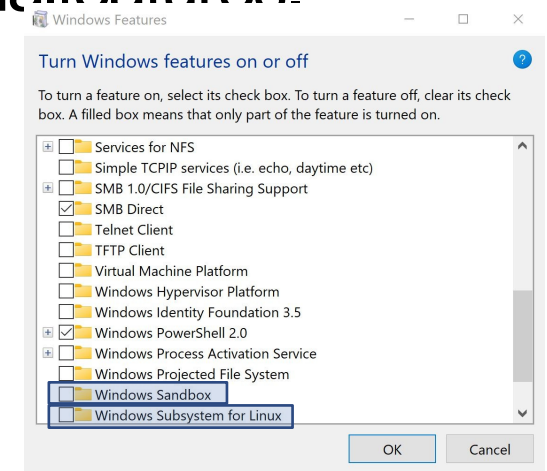


POSSIBILI FONTI DI INCOMPATIBILITÀ

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

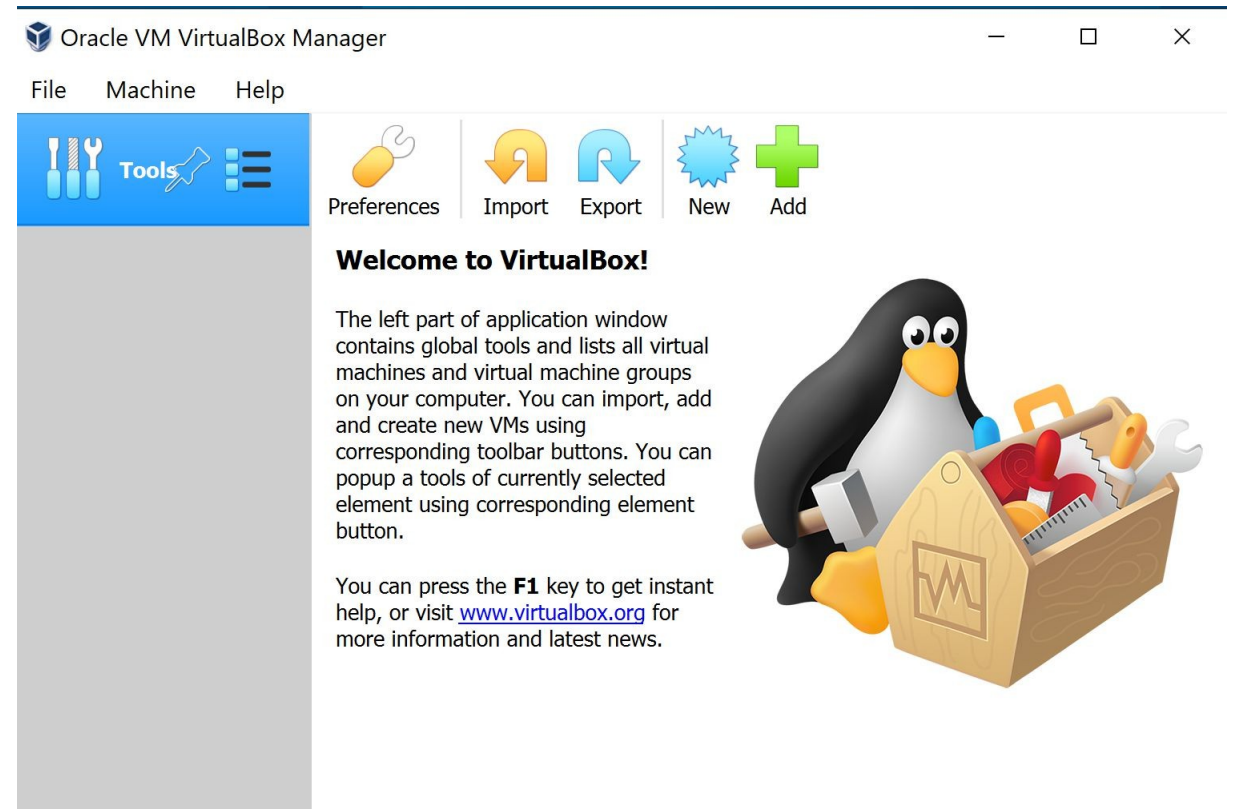
Per disattivare questi servizi, attivare la finestra Run/Esegui premendo la combinazione di tasti +R e digitare `optionalfeatures`. Cercare i 3 nomi sopra indicati e togliere il tick.

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

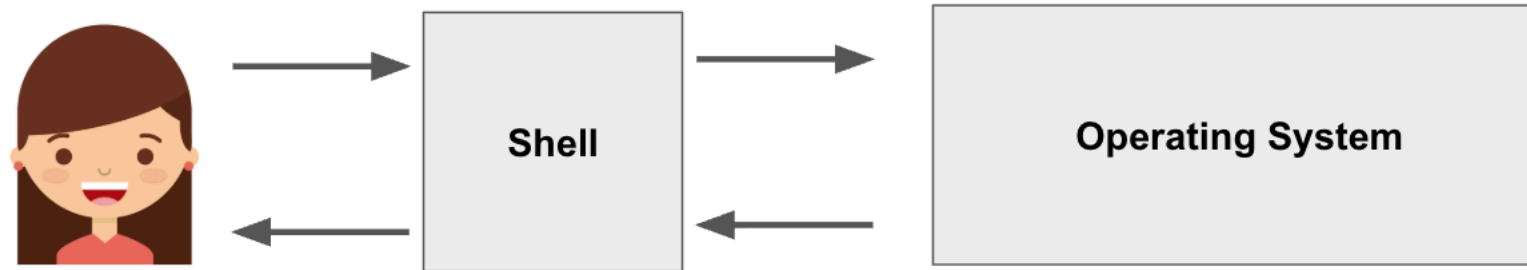
Interfaccia

4. LINUX 101

Interfaccia a linea di comando

Cosa è una shell?

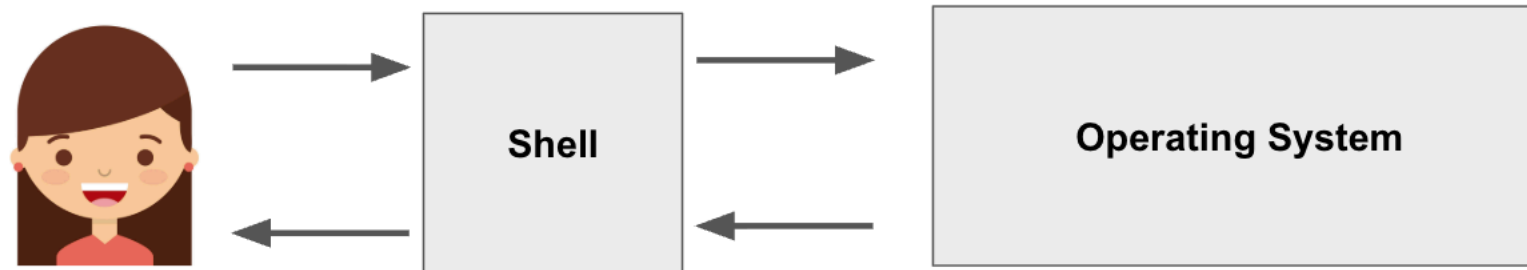
In informatica, una shell è un programma che mette a disposizione dell'utente o di altri programmi i servizi del sistema operativo.



Perchè si chiama shell? Perchè è lo strato più esterno del sistema operativo

Cosa è una shell?

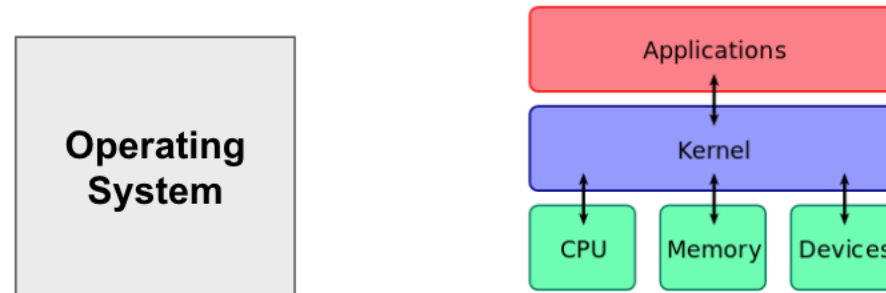
In informatica, una shell è un programma che mette a disposizione dell'utente o di altri programmi i servizi del sistema operativo.



Beh, ma perchè dovrete voler interagire con il sistema operativo? Cosa significa?

Cosa è una shell?

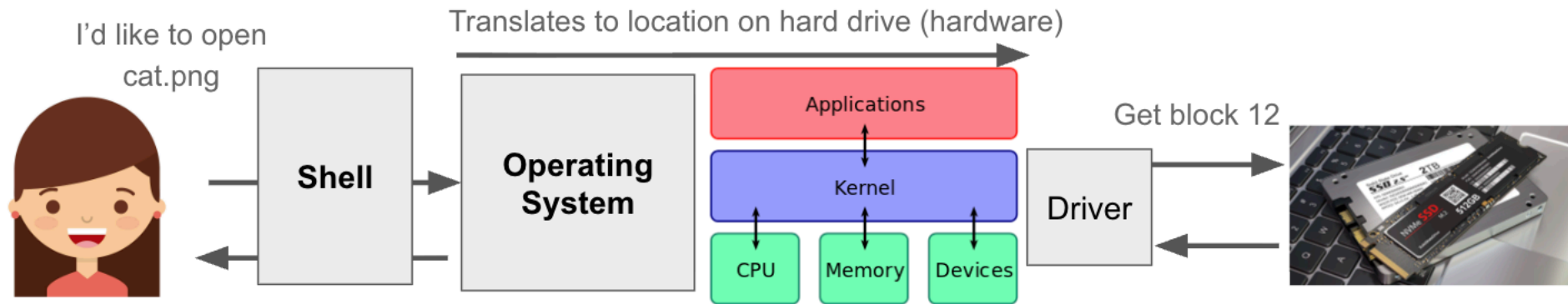
Se vuoi fare qualsiasi cosa con il tuo computer, stai interagendo con il sistema operativo, che tu ne sia consapevole o meno.



Il sistema operativo gestisce le risorse software e hardware del tuo computer. Pianifica il lavoro sulle CPU, assegna memoria ai processi e gestisce il file system e i dispositivi di rete.

Cosa è una shell?

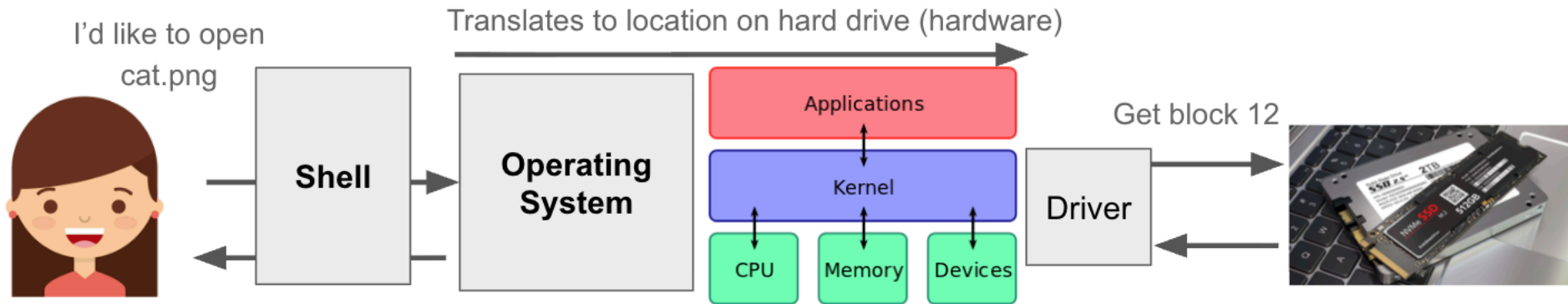
Se vuoi fare qualsiasi cosa con il tuo computer, stai interagendo con il sistema operativo, che tu ne sia consapevole o meno.



Quando apri un file, il sistema operativo individua dove si trova il file sul disco rigido e recupera i dati richiesti per te.

Cosa è una shell?

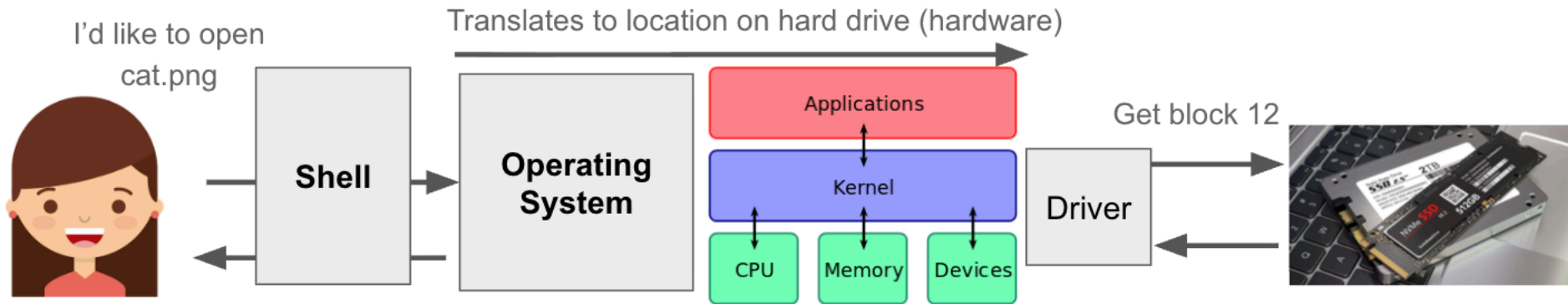
Se vuoi fare qualsiasi cosa con il tuo computer, stai interagendo con il sistema operativo, che tu ne sia consapevole o meno.



Quando esegui un programma, il sistema operativo assegna memoria al programma e lo avvia su un processore che lo esegue.

Cosa è una shell?

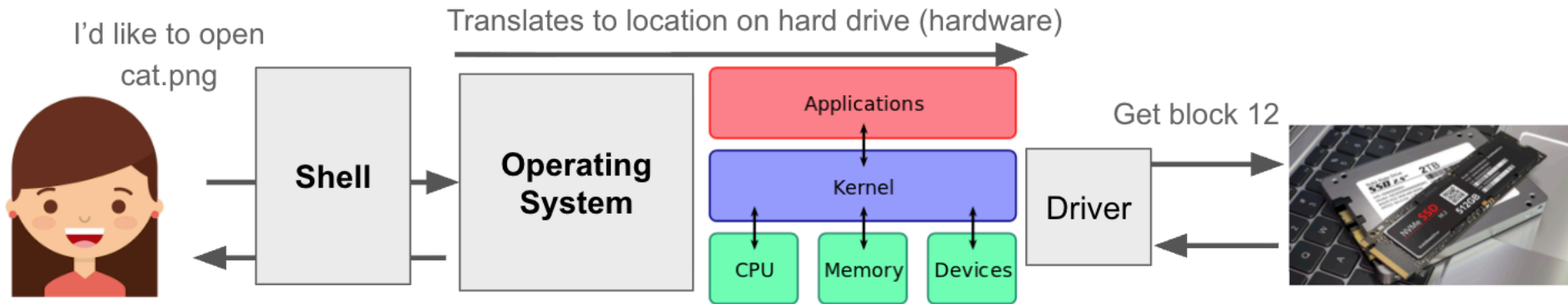
Se vuoi fare qualsiasi cosa con il tuo computer, stai interagendo con il sistema operativo, che tu ne sia consapevole o meno.



Durante l'esecuzione di un programma, il kernel passa da un programma all'altro in modo che la tua macchina rimanga reattiva. È per questo che puoi continuare a usare il computer anche se ha un solo core della CPU.

Cosa è una shell?

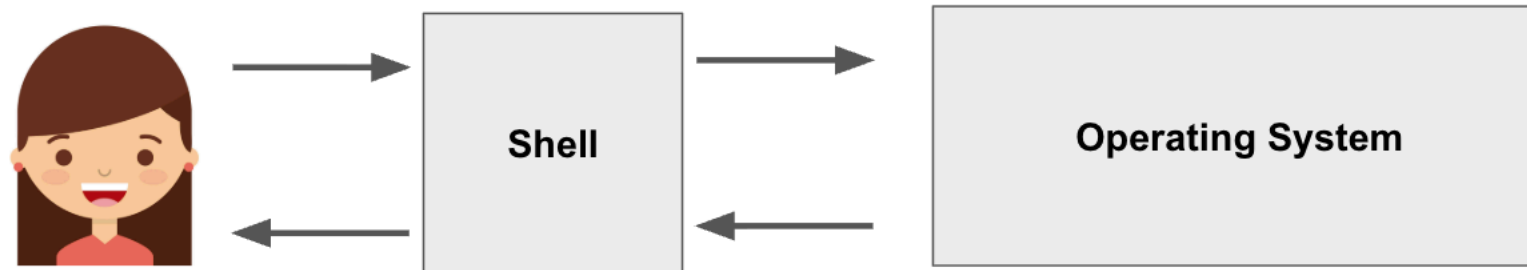
Se vuoi fare qualsiasi cosa con il tuo computer, stai interagendo con il sistema operativo, che tu ne sia consapevole o meno.



Quando accedi a una pagina web, il sistema operativo coordina la richiesta e la ricezione dei dati tramite la tua scheda di rete e gestisce in modo trasparente eventuali perdite di dati.

Cosa è una shell?

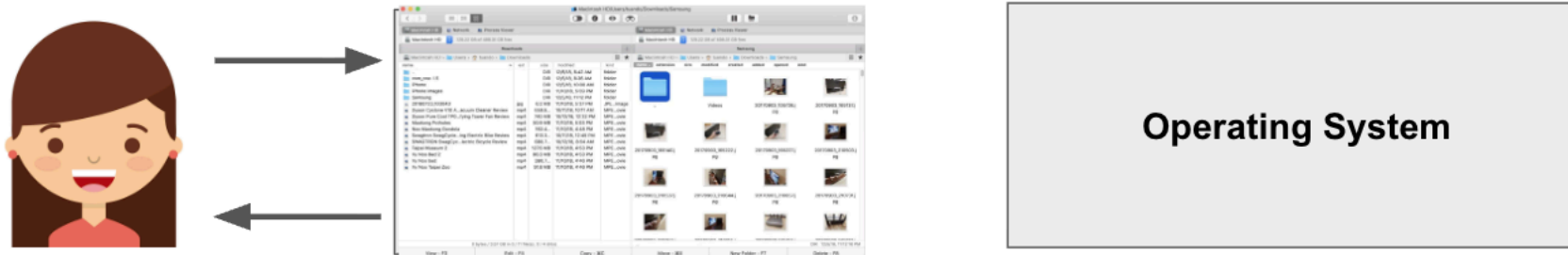
Ok, mi hai convinto. Voglio interagire con il sistema operativo. Come faccio a farlo con una shell?



Probabilmente lo avete già fatto!

Cosa è una shell?

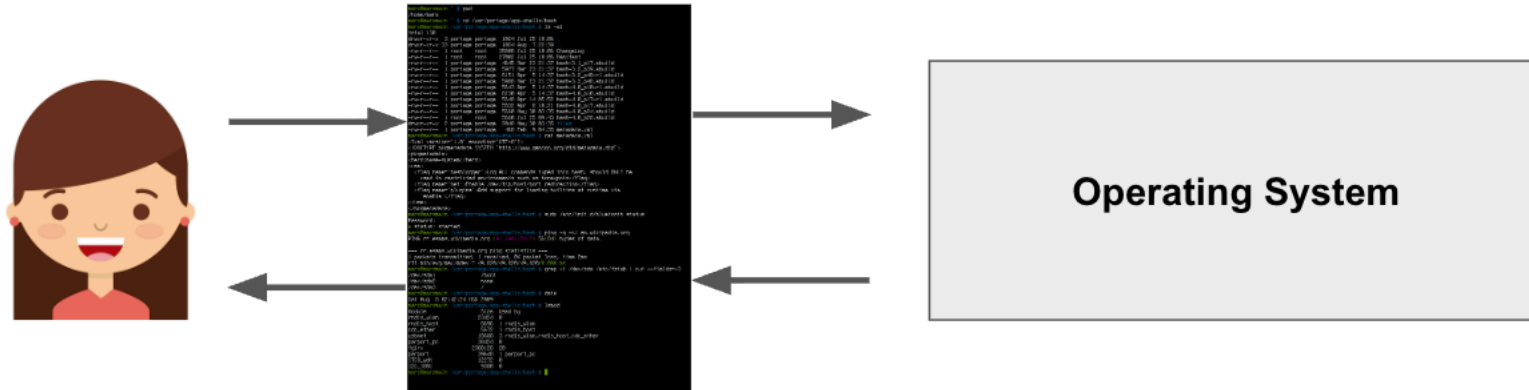
Molti programmi shell offrono **interfacce grafiche**. Probabilmente hai già usato una shell grafica in passato!



Esempi di shell grafiche: ambienti desktop, esploratori di file. Puoi interagire con il sistema operativo (leggere/scrivere file, eseguire programmi) tramite una shell grafica.

Cosa è una shell?

In questa lezione impareremo a usare programmi shell che offrono interfacce a riga di comando.



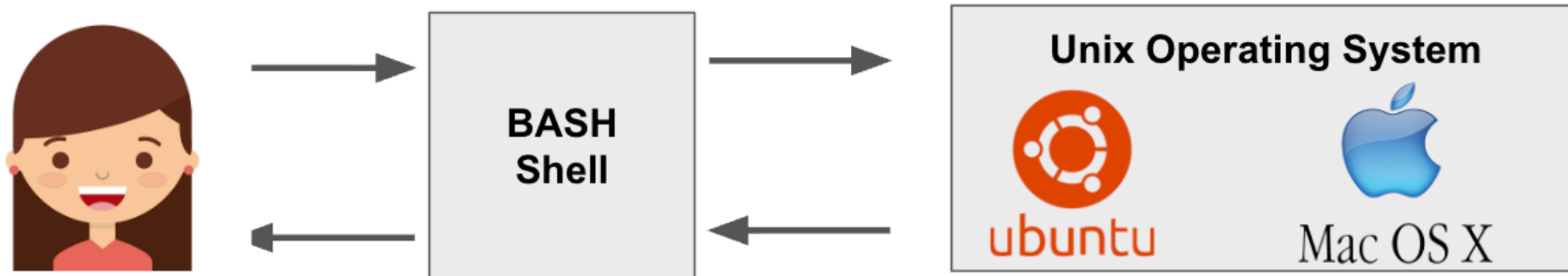
Cosa è una shell?

Perché la riga di comando? Ci sono molti motivi!

- Permette di lavorare in modo efficiente con la tastiera invece che con il mouse
- È possibile automatizzare facilmente i compiti scrivendo script
- Alcune macchine, come i server e i sistemi embedded, ti offrono solo una shell a riga di comando con cui lavorare

Introducing Bourne Again Shell (BASH)

La Bourne Again Shell (BASH) è una shell a riga di comando comunemente utilizzata nei sistemi operativi basati su Unix, come Ubuntu e Mac OSX.



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

WSL - Windows Subsystem for Linux

WSL è uno strumento che permette di usare Linux direttamente su Windows, senza dover installare una macchina virtuale o fare dual boot

- Permette di seguire il corso con gli stessi comandi Linux/Ubuntu
- Facile da installare e leggero
- Integra terminale Linux e file di Windows nello stesso ambiente

WSL - Windows Subsystem for Linux

Installazione WSL

- Apri **PowerShell** come amministratore
- Digita `wsl —install`
- Riavvia il computer se richiesto
- Alla prima apertura, scegli la distribuzione e imposta nome utente e password

Dopo l'installazione, potrai aprire il terminale Linux scrivendo `wsl`

Git Bash

Terminale che porta i comandi base di Linux/Unix su Windows. Gratuito, leggero e facile da usare.

- Vai sul sito ufficiale <https://gitforwindows.org>
- Clicca su Download e scarica il file di installazione
- Avvia il programma lasciando tutte le impostazioni predefinite
- Dopo l'installazione, apri **Git Bash** dal menu start

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

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

Pwd

- Mostra il percorso assoluto della cartella in cui ti trovi attualmente
- Utile per sapere sempre dove sei nel filesystem

LS – LIST FILES

ls

- Mostra in output la lista dei file e delle cartelle contenuti in una directory
- Può essere usato:
 - senza argomento → mostra i contenuti della cartella corrente
 - con argomento → mostra i contenuti della cartella specificata

CD – CHANGE DIRECTORY

cd

- Richiede il passaggio di un argomento: nome della cartella di destinazione (**cd Desktop**)
- Dobbiamo conoscere la struttura della working directory per muoverci

Ora che sappiamo spostarci, controlliamo con pwd come cambia il percorso assoluto cambiando directory.

PICCOLO EXCURSUS SULLE CARTELLE DI SISTEMA: LA CARTELLA RADICE

E la radice del filesystem: da qui partono tutte le altre cartelle.

Contiene directory di sistema e di configurazione:

- `/bin` → comandi di base (es. `ls`, `cp`, `mv`)
- `/etc` → file di configurazione del sistema
- `/usr` → programmi e librerie installate
- `/var` → file variabili (log, cache, ecc.)
- `/tmp` → file temporanei
- `/dev` → dispositivi hardware (es. dischi, tastiera)
- `/home` → cartelle personali degli utenti

PICCOLO EXCURSUS SULLE CARTELLE DI SISTEMA: LA CARTELLA home

Dentro home ogni utente ha la sua propria cartella personale.

- `/home/studente`
- Qui trovi i tuoi documenti, Desktop, Download etc
- La scorciatoia `~` porta direttamente alla tua home personale

CD – CHANGE DIRECTORY

Alcune cartelle del filesystem hanno dei simboli speciali:

- `/` Questo simbolo rappresenta la radice del filesystem, il punto di partenza di tutto ciò che esiste sul sistema. Tutte le directory e i file “ramificano” da qui
- `.` (punto singolo) Indica la cartella corrente. Fa riferimento alla directory in cui ti trovi.
- `..` (due punti) Fa riferimento alla cartella superiore (parent directory). Serve per salire di un livello nella struttura ad albero delle directory.
- `~` (tilde) È un'abbreviazione che rappresenta la home directory dell'utente, ovvero la directory personale (ad esempio `/home/tuo-utente` su Linux).

CD – CHANGE DIRECTORY

Inoltre, esiste una distinzione tra path assoluto e path relativo di un filesystem

Esempio, path per arrivare a Sottocartella1:

- **Path assoluto.** Parte sempre dalla **radice / del filesystem**, e indica sempre l'indirizzo completo (esempio: /home/studente/Desktop/Cartella1/Sottocartella1)
- **Path relativo.** Parte dalla cartella in cui ti trovi
 - Se ti trovi già su Desktop: Cartella1/Sottocartella1
 - Se ti trovi dentro Cartella 1: Sottocartella1

CD – IN PRATICA

- Creiamo con l'interfaccia grafica una cartella sul Desktop che si chiama "Cartella1"
- Creiamo dentro questa Cartella una cartella che si chiama "Sottocartella1"
- Da terminale, con **cd**, entriamo nella sottocartella "Sottocartella1"
- Ora torniamo con **cd** nella cartella "Cartella1". Ci riusciamo?

CD – ARGOMENTI COMPOSITI

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

Esempio:

- Spostiamoci nel Desktop
- Spostiamoci nella cartella “Sottocartella1” direttamente dal Desktop

CD – ARGOMENTI COMPOSITI

Come argomento di `cd` possiamo anche indicare il path assoluto

Che comando possiamo utilizzare per conoscere il path assoluto? Proviamo ad entrare in Sottocartella1 utilizzando il path assoluto

ALTRI CARATTERI RISERVATI IN BASH

Ci sono altri caratteri riservati per la bash:

- * Rappresenta una qualsiasi sequenza di caratteri (anche vuota)
 - `ls *.txt` mostra tutti i file che finiscono con .txt
 - `ls *1` mostra tutti i file/cartelle che finiscono con 1
- ? Rappresenta un singolo carattere qualsiasi
 - `ls *.txt` mostra tutti i file con un solo carattere prima di .txt
 - `ls ?.txt` mostra file1.doc, fileA.doc, ma non file12.doc

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*.



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



Utilizzato per separare i nomi di cartelle e file nei percorsi

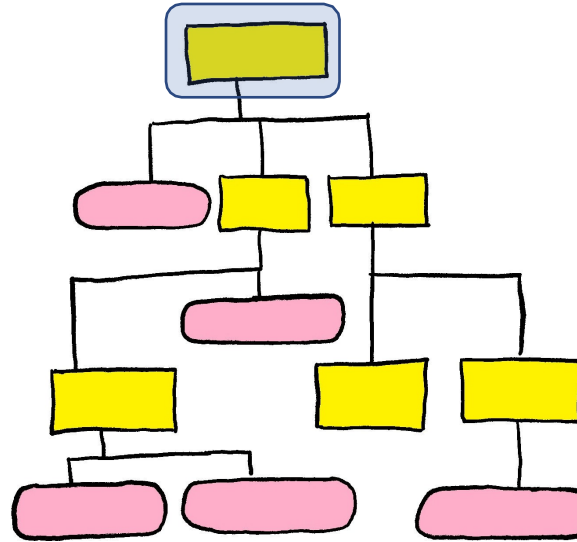
LA STRUTTURA DEL FILESYSTEM DI UBUNTU

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
```

MKDIR – CREAZIONE CARTELLE

`mkdir` (makedirectory) serve a creare una cartella vuota

- Parametro: percorso (relativo o assoluto) + nome cartella
- NON METTERE SPAZI NEL NOME UNA CARTELLA!!!

Esercizio: Creare (da terminale) una nuova cartella dentro "Cartella1", chiamata "Sottocartella2"

TOUCH – CREAZIONE FILE VUOTI

`touch` serve a creare un nuovo file vuoto

- Parametro: percorso (relativo o assoluto) + nome file
- Se il file esiste già `touch` ne aggiorna la data/ora di modifica

Esercizio: Creare (da terminale) una file vuoto chiamato "prova.txt" dentro "Sottocartella2"

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.



Riassunto comandi

- pwd → print working directory
- ls → list
- mkdir → make directory
- touch → create empty file / update timestamp
- cd → change directory
- . → current directory
- .. → parent directory
- ~ → home directory

IL COMANDO MAN

man

- Mostra il manuale di un comando
- Utile per capire:
 - cosa fa il comando
 - quali opzioni e **parametri** accetta
 - esempi di utilizzo

Esercizio: Andiamo ad usare man con il comando ls, come facciamo ad eseguire il comando ls ricorsivamente su tutte le subdirectory?

ALTRI CARATTERI RISERVATI IN BASH

Possiamo creare più file con touch, semplicemente scrivendo tutti i file che vogliamo creare di seguito

```
touch appunti1.txt appunti2.txt appunti3.txt
```

Poi se vogliamo mostrare tutti i file che iniziano con appunti

```
ls appunti*
```


ALTRI CARATTERI RISERVATI IN BASH

Ma se vogliamo creare 15 file di appunti, esiste un modo ottimale?

```
touch appunti{1..15}.txt
```

Oppure se vogliamo creare più file .txt con nomi diversi

```
touch {lezione,esercizio,soluzione}.txt
```

ALTRI CARATTERI RISERVATI IN BASH

La stessa logica funziona per mkdir

```
mkdir dir{1..15}
```

Prima di proseguire, un trucchetto

Richiamare l'ultimo comando

- Nella shell, la freccia in su sulla tastiera permette di:
 - Richiamare l'ultimo comando eseguito
 - Scorrere la cronologia dei comandi precedenti
- In modo simile, la freccia in giù permette di andare avanti nella cronologia dei comandi (torno a comandi più recenti dopo aver usato la freccia in su)

Prima di proseguire, un trucchetto

Ma possiamo fare di più

- `Ctrl + R`:
 - Cerca nella cronologia dei comandi
 - Digita una parte del comando → la shell propone il più recente che corrisponde
 - Premi di nuovo `Ctrl + R` per scorrere altri risultati

CP – COPIA FILE

cp

1. Serve a copiare file o cartelle
2. Richiede due argomenti (sorgente e destinazione):
 - `cp path_sorgente path_destinazione`

Esercizio:

1. Vai nella cartella "Cartella1"
2. Crea un file vuoto chiamato "prova.txt"
3. Copialo dentro "Sottocartella1"
4. Verifica di aver eseguito la copia correttamente

CP – RINOMINAZIONE DEL FILE

Copiare e rinominare con cp

- E possibile rinominare un file mentre lo si copia
- Basta specificare il nuovo nome come parte del percorso di destinazione

Esercizio:

1. Vai nella cartella "Cartella1"
2. Crea un file vuoto chiamato "appunti.txt"
3. Copialo dentro "Sottocartella1" rinominandolo "appunti_nuovo.txt"
4. Verifica di aver eseguito la copia correttamente

CP – COPIA NELLA STESSA CARTELLA

Ammattiamo ora di voler creare una copia di appunti.txt all'interno di Cartella1 (chiamato "appunti2.txt") in modo da poterla modificare tenendo una copia dell'originale.

Come faccio?

NOTA – ESTENSIONE DEL FILE

Nei sistemi Unix è possibile creare un file anche senza estensione:

- Provate a creare un file senza estensione

In Bash esiste il comando `file` che prova a determinare il tipo di un file anche se non ha l'estensione

CP – COPIA CARTELLE

Proviamo ora a copiare una cartella

- Entriamo dentro Cartella1
- Proviamo a copiare "Sottocartella1" dentro "Sottocartella2"

Non funziona! Perché le cartelle vogliono anche il flag `-r` (ricorsivo)

- `cp -r Sottocartella1 Sottocartella2/`

MV – MUOVI FILE

mv

- Serve a spostare/rinominare file o cartelle
- Richiede due argomenti (sorgente e destinazione):
 - mv path_sorgente path_destinazione

Esercizio:

1. Vai nella cartella "Cartella1"
2. Crea un file vuoto chiamato "prova_mv.txt"
3. Spostalo dentro "Sottocartella1"
4. Verifica di aver eseguito lo spostamento correttamente

MV – MUOVI FILE

Se vogliamo muovere una intera cartella? Come facciamo?

MV – MUOVI FILE


Abbiamo detto che serve anche a rinominare un file.
Per rinominare un file posso muovere il file originale nello stesso luogo ma cambiando il suo nome

Se voglio rinominare il file prova.txt lasciandolo nella stessa posizione del filesystem quello che dovrò fare sarà:

```
mv prova.txt nuovo_nome.txt
```

RM – RIMUOVI FILE O CARTELLA

rm

- Serve a cancellare file o cartelle
- ATTENZIONE: I FILE ELIMINATI NON VANNO NEL CESTINO MA SONO PERSI PER SEMPRE 
- Eliminare un file: `rm file.txt`
- Eliminare una cartella: `rm -r Cartella`
- Chiedere conferma prima di cancellare: `rm -i file.txt`

RIMUOVERE FILE O CARTELLE DI SISTEMA

Possiamo rimuovere file o cartelle di sistema con `rm` o `rmdir`?

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)

I comandi cp, mv, rm possono anche lavorare su più file o cartelle usando caratteri speciali * e ?

Esempio1: se vogliamo copiare tutti i file .txt in una cartella

```
cp *.txt Sottocartella1/
```

Esempio2: vogliamo eliminare tutte le estensioni png

```
rm *.png
```


ESERCIZIO INSIEME

- Vai nella tua home directory
- Crea una cartella chiamata `Esercizio` e dentro una sottocartella `Test`
- Crea 3 file vuoti: `prova1.txt`, `prova2.txt`, `prova3.txt`
- Copia tutti i file `*.txt` dentro `Test`
- Sposta `prova1.txt` dalla cartella corrente alla tua home directory
- Rinomina `prova2.txt` in `esempio.txt`
- Cancella `prova3.txt`
- Vai dentro `Test` ed esplora con `man rm` le opzioni disponibili
- Torna in `Esercizio` ed elimina la cartella `Test`



Riassunto comandi

- `pwd` → print working directory
- `ls` → list
- `mkdir` → make directory
- `touch` → create empty file / update timestamp
- `cd` → change directory
- `.` → current directory
- `..` → parent directory
- `~` → home directory



Riassunto comandi

- `man` → manual
- `cp` → copy
- `mv` → move
- `rm` → remove

ALTRI COMANDI BASH

Scaricare da GitHub il *.txt

ALTRI COMANDI BASH – ECHO

echo Serve a stampare un messaggio su una stringa o su un terminale

1. `echo "Ciao Mondo!"` Per stampare su stringa

2. `echo "Ciao Mondo!" >> file.txt` Per stampare su file

Sequenze di escape comuni

- `\n` → nuova linea
- `\t` → tabulazione

ALTRI COMANDI BASH – LEGGERE FILE DA SHELL

cat Mostra il contenuto di un file tutto insieme

```
cat note.txt
```

less Mostra il file una pagina alla volta, permette di scorrere

```
less note.txt
```

head Mostra solo le prime righe di un file (di default 10)

```
head note.txt
```

```
head -n 5 note.txt
```

tail Mostra solo le ultime righe di un file (di default 10)

```
tail note.txt
```

```
tail -n 5 note.txt
```

MANIPOLAZIONE DI DATI

sort Serve per ordinare le righe di un file

1. `sort dati_estesi.txt` Ordina alfabeticamente

2. `sort -n -t ',' -k 3 dati_estesi.txt` Ordina per età (colonna 3)

3. `sort -r -t ',' -k 2 dati_estesi.txt` Ordina per cognome in ordine inverso

MANIPOLAZIONE DI DATI

wc Conta linee, parole e byte

Opzioni principali:

- `-l` → linee
- `-w` → parole
- `-c` → byte

```
1.wc dati_estesi.txt
```

```
2.wc -l dati_estesi.txt
```

```
3.wc -w dati_estesi.txt
```

```
# Linee, parole, byte
```

```
# Numero di linee
```

```
# Numero di parole
```

MANIPOLAZIONE DI DATI

cut Estrae colonne specifiche da file tabellari

Opzioni principali:

- `-f` → colonne da prendere
- `-d` → delimitatore (di default TAB, qui usiamo ,)

```
cut -f 3,6 -d ',' dati_estesi.txt
```

MANIPOLAZIONE DI DATI

`grep` Cerca righe che contengono un pattern

Opzioni utili:

- `-v` → mostra le righe che non contengono il pattern

1. `grep "Trieste" dati_estesi.txt` # Tutti quelli che vivono a Trieste
2. `grep -v "Torino" dati_estesi.txt` # Tutti tranne quelli di Torino
3. `grep "^A" dati_estesi.txt` # Tutti i nomi che iniziano con A

SCRIVERE SU .TXT CON VIM

vim Editor di testo (origini risalenti al 1976)

Opzioni utili:

- `:w` → salva il contenuto di un file
- `:q` → esce da vi
 - `:q!` → forza l'uscita (senza salvare)
 - `:wq` → esco e salvo
- `:i` → entra in modalità inserimento
- `:dd` → cancella la riga in cui si trova il cursore
- `:dw` → cancella la parola in cui si trova il cursore
- `ESC` → esce dalla modalità inserimento

ESERCIZIO INSIEME

1. Conta quante righe ci sono nel file.
2. Conta quante parole ci sono in tutto il file.
3. Visualizza le prime 3 righe e salvale in un nuovo file chiamato `prime3.txt`.
4. Visualizza le ultime 2 righe e salvale in un nuovo file `ultime2.txt`.
5. Ordina alfabeticamente tutte le righe e salvale in `ordinato.txt`.
6. Ordina le righe in ordine inverso e visualizza solo a schermo.
7. Estrai solo la prima parola di ogni riga.
8. Cerca tutte le righe che contengono la parola `shell`.
9. Cerca tutte le righe che non contengono la parola `Linux`.
10. Apri il file in vim e aggiungi in fondo la riga: "Questa è una nuova riga di prova"
11. Verifica che il file abbia una riga in più

ESERCIZIO INSIEME

1. Conta quante righe ci sono nel file.
2. Conta quante parole ci sono in tutto il file.
3. Visualizza le prime 3 righe e salvale in un nuovo file chiamato `prime3.txt`.
4. Visualizza le ultime 2 righe e salvale in un nuovo file `ultime2.txt`.
5. Ordina alfabeticamente tutte le righe e salvale in `ordinato.txt`.
6. Ordina le righe in ordine inverso e visualizza solo a schermo.
7. Estrai solo la prima parola di ogni riga.
8. Cerca tutte le righe che contengono la parola `shell`.
9. Cerca tutte le righe che non contengono la parola `Linux`.
10. Apri il file in `vim` e aggiungi in fondo la riga: "Questa è una nuova riga di prova"
11. Verifica che il file abbia una riga in più

```
1.wc -l appunti.txt
```

```
2.wc -w appunti.txt
```

```
3.head -n 3 appunti.txt > prime3.txt
```

```
4.tail -n 2 appunti.txt > ultime2.txt
```

```
5.sort appunti.txt > ordinato.txt
```

```
6.sort -r appunti.txt
```

```
7.cut -d ' ' -f 1 appunti.txt
```

```
8.grep "shell" appunti.txt
```

```
9.grep -v "Linux" appunti.txt
```

```
10.vim appunti.txt
```

```
11.wc -l appunti.txt
```

ALTRI COMANDI BASH – VISUALIZZAZIONE PERMESSI

ls -l Mostra i permessi dei file nella directory corrente

- Categorie
 - u → proprietario (user)
 - g → gruppo (group)
 - o → altri (others)
- Permessi
 - r → read (lettura)
 - w → write (scrittura)
 - x → execute (esecuzione)
- Struttura
 - Primi 3 caratteri → proprietario (u)
 - Successivi 3 caratteri → gruppo (g)
 - Ultimi 3 caratteri → altri utenti (o)

-rwxr-xr--

ALTRI COMANDI BASH – MODIFICA PERMESSI

chmod Modifica permessi dei file e delle cartelle

```
chmod [chi] [operatore] [permessi] file
```

- Chi: u (user/proprietario), g (gruppo), o (altri), a (tutti)
- Operatore: + (aggiungi), - (rimuovi), = (imposta esattamente)
- Permessi: r (read), w (write), x (execute)
- Esempi
 - `chmod u+x script.sh` # aggiunge permesso di esecuzione al proprietario
 - `chmod g-w file.txt` # rimuove permesso di scrittura al gruppo
 - `chmod a=r file.txt` # tutti possono solo leggere
- Sintassi Ottale
 - Ogni permesso ha un valore numerico: r = 4, w = 2, x = 1
 - Si sommano i valori: 7 = rwx, 6 = rw-, 5 = r-x, 4 = r—
 - `chmod 755 script.sh` -> proprietario = 7 (rwx), gruppo = 5 (r-x), altri = 5 (r-x)

ESERCIZIO INSIEME

1. Vai nella tua home e crea una nuova cartella `permessi`
2. Crea un file `segreto.txt` e scrivici dentro un messaggio
3. Verifica i permessi iniziali del file
4. Rimuovi a tutti il permesso di lettura
5. Prova a leggerlo con `cat`
6. Ridai al proprietario il permesso di lettura
7. Controlla di nuovo i permessi e verifica che adesso tu possa leggere il file

ESERCIZIO INSIEME

1. Vai nella tua home e crea una nuova cartella `permessi`
2. Crea un file `segreto.txt` e scrivici dentro un messaggio
3. Verifica i permessi iniziali del file
4. Rimuovi a tutti il permesso di lettura
5. Prova a leggerlo con `cat`
6. Ridai al proprietario il permesso di lettura
7. Controlla di nuovo i permessi e verifica che adesso tu possa leggere il file

```
1.cd ~      mkdir permessi    cd permessi
2.echo "Questo è un file segreto!" > segreto.txt
3.ls -l segreto.txt
4.chmod a-r segreto.txt
5.cat segreto.txt
6.chmod u+r segreto.txt
7.ls -l segreto.txt  cat segreto.txt
```

Processi

Un processo è un programma in esecuzione.

Ogni volta che lanci un comando o apri un'app, il sistema crea un processo.

Un processo ha:

- **PID** (Process ID) → numero che lo identifica
- **Stato** → in esecuzione, in attesa, terminato...
- **Risorse** → CPU, memoria, file aperti...

I processi sono gestiti dal **kernel** che decide come e quando eseguirli.

Vedere i processi e le specifiche del PC

- Elencare i processi attivi **ps aux** mostra tutti i processi in corso con utente, PID, uso CPU/memoria
- Processi in tempo reale **top** una “finestra live” con consumo di CPU e RAM
- Uscire da top → premi q

Processi

Esercizio: Terminare un processo (con vim)

1. Apri un file con vim (questo sarà il processo da terminare):
`vim prova.txt`
2. In un altro terminale, elenca i processi in esecuzione:
`ps aux | grep vim`
Vedrai una riga con `vim prova.txt` e il suo PID (seconda colonna).
3. Termina il processo:
`kill <PID>`
4. Torna al primo terminale: noterai che vim si è chiuso bruscamente e sei tornato alla shell.

ALTRI COMANDI BASH – CARATTERISTICHE DEL PC

Numero di CPU

- nproc (Linux)
- sysctl -n hw.ncpu (MacOS)

Dettagli sulla CPU

- lscpu (Linux)
- sysctl -n machdep.cpu.brand_string (MacOS)

Memoria disponibile

- free -h (Linux)
- vm_stat (MacOS)

Versione del kernel e del sistema operativo

- uname -a (Linux)
- sw_vers (MacOS)

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`

Uso di **sudo** con i package manager

- Perché serve?
 - Installare o rimuovere programmi modifica il sistema
 - Queste operazioni richiedono i permessi da amministratore
- Come si fa?
 - Si antepone sudo al comando
 - Il sistema chiede la password dell'utente

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
 - `sudo apt update`
 - **Descrizione:** Aggiorna l'elenco dei pacchetti disponibili nel repository.
2. Aggiornare i pacchetti installati
 - `sudo apt upgrade`
 - **Descrizione:** Aggiorna tutti i pacchetti installati alle versioni più recenti disponibili.
3. Installare un pacchetto
 - `sudo apt install <nome_pacchetto>`
 - **Descrizione:** Installa un pacchetto specifico dal repository.
4. Rimuovere un pacchetto
 - `sudo apt remove <nome_pacchetto>`
 - **Descrizione:** Rimuove un pacchetto dal sistema.

Altri comandi utili

1. Pulizia dei pacchetti non necessari
 - `sudo apt autoremove`
 - **Descrizione:** Rimuove i pacchetti inutilizzati che sono stati installati automaticamente come dipendenze.
2. Rimuovere i pacchetti scaricati
 - `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
 - `apt show <nome_pacchetto>`
 - **Descrizione:** Mostra informazioni dettagliate su un pacchetto.

Per MacOS, `sudo apt = brew`

APT E UBUNTU SOFTWARE

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

- `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.

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.

Esempio di utilizzo di `curl`

- `curl https://raw.githubusercontent.com/gpietrop/PreCorso-INF-uniTS/main/scaricami.txt`
- Questo comando scarica e mostra il contenuto della pagina web nel terminale.

Esempio pratico: Scaricare un file

- `curl -O https://raw.githubusercontent.com/gpietrop/PreCorso-INF-uniTS/main/scaricami.txt`
- **Descrizione:** Scarica il file "file.txt" dalla pagina web e lo salva nella directory corrente.

I FILE BASH

- Finora abbiamo dato comandi uno alla volta nel terminale.
- A volte però vogliamo che il computer esegua più comandi in sequenza, senza doverli riscrivere ogni volta.
- Possiamo scrivere questi comandi dentro a un file di testo.
- Questo file prende il nome di script Bash.

file.sh

I FILE BASH:

Struttura di base

- Uno script è un **file di testo** con dentro comandi della shell.
- Per convenzione, i file hanno estensione `.sh`.
- La prima riga spesso è:

`file.sh`

```
#!/bin/bash
```

si chiama **shebang**, dice al sistema che deve usare Bash per eseguire il file.

Le variabili

Le variabili in programmazione:

- Una variabile è come una scatola con un'etichetta
- Dentro ci possiamo mettere un valore (numero testo ecc)
- Ogni volta che usiamo il nome della variabile, il programma prende il valore della scatola

```
x = 5  
y = 10  
z = x + y
```

- x contiene 5
- y contiene 10
- z conterrà 15

I FILE BASH:

Struttura di base

In Bash, `$` serve per espandere una variabile (cioè leggere il suo valore).

```
NOME="Alice"  
ETA=20  
echo "$NOME ha $ETA anni"
```

I FILE BASH:

Primo Esempio

- Creiamo un file di testo
 - `vim ciao.sh`
- Scriviamo dentro al file di testo
 1. `echo "Ciao, questo è il mio primo script!"`
 2. `ls` (in una nuova riga)
- Salviamo ed eseguiamo
 - `bash ciao.sh`
 - OPPURE:
 - (Prima rendiamo lo script eseguibile) `chmod +x prova.sh`
 - `./prova.sh`

`file.sh`

ESERCIZIO INSIEME

1. Crea un file chiamato `setup.sh`.

1. `vim setup.sh`

2. Dentro lo script fai in modo che:

2. Contenuto del file

1. venga stampato un messaggio di benvenuto;

3. `chmod +x setup.sh`

2. venga creata una cartella chiamata `progetto`;

4. `./setup.sh`

3. dentro `progetto` vengano create due sottocartelle: `dati` e `report`;

4. dentro `dati` vengano creati 3 file vuoti: `input1.txt`, `input2.txt`,
`input3.txt`;

5. dentro `report` venga creato un file `log.txt` con scritto "Progetto
inizializzato con successo";

6. al termine, la shell mostri il contenuto della cartella `progetto` e delle
sottocartelle.

3. Rendi lo script eseguibile.

4. Esegui lo script e verifica che abbia fatto tutto automaticamente

Controllo del flusso

FOR serve per ripetere comandi

```
for VAR in lista; do
    # comandi
done
```

```
#!/bin/bash

# Creo una lista di nomi di file
LISTA="a b c"

# Uso la variabile LISTA nel ciclo
for NOMEFILE in $LISTA; do
    echo "Sto creando il file $NOMEFILE.txt"
    touch "$NOMEFILE.txt"
done
```

Controllo del flusso

IF serve per prendere decisioni

```
if [ condizione ]; then
    # comandi se la condizione è vera
else
    # comandi se la condizione è falsa
fi
```

```
#!/bin/bash

# Lista di file
LISTA="a b c"

for NOME in $LISTA; do
    touch "$NOME.txt"
    if [ "$NOME" = "b" ]; then
        echo "Ho trovato il file speciale: $NOME.txt"
    else
        echo "Questo è un file normale: $NOME.txt"
    fi
done
```

Controllo del flusso

WHILE ripetere finché una condizione è vera

```
while [ condizione ]; do  
    # comandi  
done
```

```
#!/bin/bash  
  
RISPOSTA="si"  
  
while [ "$RISPOSTA" = "si" ]; do  
    echo "Vuoi continuare? (si/no)"  
    read RISPOSTA  
done  
  
echo "Hai scelto di fermarti."
```

Controllo del flusso

1. Nella tua home, crea una cartella `labshell` e spostati dentro.
2. Crea un file chiamato `progetto.sh`.
3. Apri il file e scrivi lo **shebang** in prima riga (`#!/bin/bash`).
4. Definisci una **variabile** `NOME_PROGETTO` con valore `demo`.
5. Stampa a schermo: `Creo il progetto $NOME_PROGETTO` (usa `echo` e la variabile).
6. Crea la cartella del progetto: `demo/` e, dentro, le sottocartelle `src/` e `docs/`.
7. Definisci una **lista** in una variabile, ad es. `FILE="a b c"`.
8. Con un **ciclo for**, crea in `src/` i file `a.txt`, `b.txt`, `c.txt`.
9. Dentro il ciclo, usa un **if**:
 - se il nome è `b`, scrivi nel file `b.txt` la riga `File speciale`;
 - altrimenti scrivi `File normale`.
10. Crea in `docs/` un file `README.txt` con la frase `Questo è il progetto demo`.
11. Copia `README.txt` in `demo/` (radice del progetto).
12. Sposta `b.txt` da `src/` a `docs/`.
13. Elimina `c.txt` da `src/`.
14. Alla fine, mostra l'intera struttura con `ls -R`.
15. Rendi lo script **eseguibile** ed eseguilo. Verifica che tutto sia stato creato come previsto.

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.