# Introduzione alla shell di Linux (Bash)

Audace Sailing Team

19 novembre 2025

# Introduzione

# Obiettivi della presentazione

- Comprendere che cos'è una shell in un sistema Linux
- Conoscere le basi di Bash e dei comandi fondamentali
- Imparare a gestire file, directory

# Cos'è la shell (e cos'è Bash)

# Shell e terminale

- La shell è il programma che accetta comandi testuali e li esegue
- È l'interfaccia tra l'utente e il sistema operativo
- Il terminale (GNOME Terminal, Konsole, ecc.) è la "finestra"
- La shell è il processo che gira dentro il terminale

# Bash in breve

- Bash = Bourne Again SHell
- Una delle shell più diffuse su sistemi Linux
- Permette di:
  - interpretare comandi
  - automatizzare attività tramite script
  - gestire variabili, redirezioni, pipe, job, ecc.

# Primi comandi fondamentali

# Struttura di un comando

- Forma generale:

  ```
  comando opzioni argomenti
  ```

- Esempio:

  ```
  ls -l /home
  ```

- Dove:
  - `ls` → comando
  - `-l` → opzione
  - `/home` → argomento

# Dove trovare aiuto

- `man comando` mostra il manuale dettagliato
- `comando --help` elenca le opzioni principali
- `help comando` (per built-in Bash)
- uso di `/` dentro `man` per cercare testo

# Comandi di base

- `pwd` → mostra la directory corrente
- `ls`, `ls -l`, `ls -a` → elenca file (anche nascosti)
- `cd /percorso` → cambia directory
- `cd` → torna alla home
- `mkdir nuova_dir` → crea directory
- `rm file.txt` → rimuove file
- `rmdir vuota` → rimuove directory vuota

# Scorciatoie utili

- `Ctrl-C`: interrompe un comando in esecuzione
- `Ctrl-L`: pulisce lo schermo
- `Ctrl-A` / `Ctrl-E`: inizio/fine riga
- `Ctrl-R`: ricerca nella cronologia dei comandi

# File system e percorsi

# Gerarchia del filesystem

- Un unico albero che parte da /
- Alcuni percorsi tipici:
  - `/home/utente` → directory personale
  - `/etc` → file di configurazione
  - `/usr` → programmi e librerie
  - `/var` → dati variabili (log, cache, . . . )

# Percorsi assoluti e relativi

- Assoluto: parte da /
  - Esempio:

```
cd /home/utente/progetto
```

- Relativo: parte dalla directory corrente
  - Esempi:

```
cd progetto      # sottodirectory
cd ..            # directory superiore
cd ../altro      # su di uno, poi giù in "altro"
```

- Simboli utili:
  - . → directory corrente
  - .. → directory superiore
  - ~ → home dell'utente

# Manipolare file e directory

# Copiare, spostare, eliminare

- Copia file:

```
cp origine.txt copia.txt
cp -r dir1 dir2        # copia ricorsiva directory
```

- Spostare / rinominare:

```
mv file.txt nuova_posizione/
mv vecchio_nome.txt nuovo_nome.txt
```

- Rimuovere:

```
rm file.txt
rm -r directory
```

# Creare file vuoti

- Creazione di un file vuoto:

```
touch file.txt
```

# Attenzione ai comandi distruttivi

- Comandi pericolosi, ad esempio:

  ```
  rm -rf /
  ```

- Possibile perdita totale di dati
- Sempre verificare due volte il comando prima di eseguirlo

# Visualizzare il contenuto dei file

# Strumenti principali

- `cat file.txt` → mostra tutto (per file piccoli)
- `less file.txt` → visualizzazione interattiva:
  - frecce per scorrere
  - `/parola` per cercare
  - `q` per uscire
- `head` / `tail`:
  - prime/ultime righe del file

# Esempi

- Prime 20 righe:

  ```
  head -n 20 file.txt
  ```

- Ultime 50 righe:

  ```
  tail -n 50 file.txt
  ```

- Seguire un log in tempo reale:

  ```
  tail -f log.txt
  ```

# Editor di testo da terminale

- Non sempre VSCode/gedit/... è disponibile, certamente non da terminale!
- Molte scelte con diversi livelli di difficoltà:
  - `nano`: semplice, sufficiente per piccole modifiche su terminale
  - `vim`: decisamente più complesso (come si esce da `vim`?)
  - `emacs`: comparabile a `vim`, molto più potente di `vim` (ha un client di posta elettronica...)

# Ricerca di testo nei file

- Cercare una stringa/espressione/... nel contenuto di un file (e non solo)
- `grep` (Global, Regular Expression, Print)
  - cercare testo in un file

    ```
    grep casa file.txt
    ```

  - cercare testo ricorsivamente nel contenuto di una cartella

    ```
    grep casa cartella/ -r
    ```

# Wildcard e globbing

# Espansioni della shell

- `*` → qualunque sequenza di caratteri
- `?` → un singolo carattere
- `[abc]` → un carattere tra quelli elencati
- `[0-9]` → una cifra

# Esempi pratici

- File di testo:

  `ls *.txt`
- File con due caratteri di suffisso:

  `ls img_??.png`
- Rimozione di log:

  `rm log_2024-*.log`
- Buona pratica: provare prima con `ls` per vedere cosa verrebbe toccato

# Cronologia comandi

# History e ricerca

- Visualizzare la cronologia:

```
history
```

- Rieseguire un comando specifico:

```
!123    # comando numero 123
```

- Strumenti interattivi:
  - frecce su/giù per scorrere
  - `Ctrl+R` per ricerca nella history

# Ottenere privilegi: sudo

# A cosa serve sudo

- Eseguire comandi con privilegi amministrativi
- Alternativa più sicura rispetto ad accedere come `root`

# Utilizzo base

- `sudo comando`: richiede la password dell'utente, non quella di root
- `sudo -u altro_utente comando`: esegue come altro utente.
- `sudo -s` o `sudo -i`: apre una shell privilegiata (sconsigliato se non necessario).

# Buone pratiche

- Usare `sudo` solo quando indispensabile
- Leggere bene il comando prima di confermare
- Evitare di usarlo in combinazione con comandi pericolosi (`rm -r`, script non verificati)
- Nota: tutti i comandi eseguiti con sudo vengono registrati nei log di sistema (importante per ambienti condivisi, *e.g.* cluster HPC)

# Esempi pratici: installazione software

- Installare un pacchetto (Debian/Ubuntu):
  - `sudo apt update`
  - `sudo apt install nomepacchetto`
- Rimuovere un pacchetto:
  - `sudo apt remove nomepacchetto`
- Aggiornare il sistema:
  - `sudo apt upgrade`

# Esempi pratici: modificare file di sistema

- Editare file di configurazione:
  - `sudo nano /etc/hosts`
  - `sudo nano /etc/fstab`
  - `sudo nano /etc/ssh/sshd_config`
- Verifica sintassi prima di riavviare servizi
  - Es.: `sshd -t` per controllare la configurazione SSH
- Mai editare file critici senza backup:
  - `sudo cp /etc/hosts /etc/hosts.backup`

# Esempi pratici: comandi amministrativi utili

- Esaminare log di sistema:
  - `sudo journalctl -xe`
  - `sudo dmesg | less`
- Montare o smontare un filesystem:
  - `sudo mount /dev/sdb1 /mnt`
  - `sudo umount /mnt`
- Ricaricare regole firewall (solo esempio generico):
  - `sudo ufw reload`

# File di configurazione

# ~/.bashrc e ~/.bash_profile

- ~/.bashrc:
  - eseguito per shell interattive
  - contiene alias, variabili, funzioni, prompt
- ~/.bash_profile (o ~/.profile):
  - eseguito al login
  - spesso richiama .bashrc:

```
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

- Buona pratica: mantenere questi file ordinati e commentati

# Buone pratiche

# Principi generali

- Non eseguire comandi da internet senza capire cosa fanno
- Limitare l'uso di `sudo` allo stretto necessario
- Usare con molta cautela `rm -rf` e simili
- Testare gli script su dati non critici prima della produzione