SHELL E TERMINALI

Nei sistemi Unix la shell è un interprete di comandi che si può usare sia in modalità interattiva sia tramite script.
Per usare il Kernel servono dei programmi utente per interagire indirettamente col kernel usa una shell.
Quando sul terminale usiamo il comando "ls" stiamo usando la shell, probabilmente anche una bash.
Kernel è il nucleo mentre la shell è il nucleo. È un programma che permette all'utente finale (umano) di parlare col kernel digitando i comandi nella tastiera. Per quanto riguarda unix la shell non ha nessun privilegio in più di qualsiasi programma che lanciamo. (shell ed elden ring possono girare allo stesso tempo).
Come interagiamo con la shell? Tramite un terminale, un terminale è un'evoluzione della telescrivente ovvero (quello fisico) è composto da una tastiera che serve per mandare i comandi al calcolatore e un dispositivo di output che ci permette di vedere ovviamente il risultato. Quello moderno invece è una finestra con cui noi interagiamo, abbiamo sempre una tastiera e un monitor (canale in cui mandiamo verso il terminale). In unix tutto è un file quindi alcuni file sono appunto dei terminal.

```
gio ~ didattica so 02_shell_e_terminali (e) py3 master start 02_shell_e_terminali.pdf
gio ~ didattica so 02_shell_e_terminali (e) py3 master tty
/dev/pts/1
gio ~ didattica so 02_shell_e_terminali (e) py3 master
```

Il dev/pts/1 sarebbe il file che corrisponde a questa finestra (il terminale). Se andassi a scrivere in quel file quello che scrive appare in questa finestra.
Su Linux ci sono due concetti simili:

- **Virtual console:** terminali che condividono la tastiera e lo schermo. Sono degli emulatori di programmi che emulano lo stato hardware.
- **Emulatore di terminale:** implementati tramite pseudoterminali, ossia file che fingono di essere un terminale tramite programma

Ogni processo usa tre file descriptor, un file descriptor è un intero che rappresenta un file aperto per il processo e ogni processo ha questi tre:

1. Standard input (stdin/cin)
2. Standard output (stdout/cout)
3. Standard error (stderr/cerr)

Lo standard Input è l'input per il processo. Gli altri due rappresentano l'output del terminale.
Lo standard output è bufferizzato ovvero che quello che scriviamo non è necessariamente scritto immediatamente sul terminale. Mentre lo standard error non lo è.

Comandi standard:

- / : radice
- /bin e /sbin : comandi essenziali e comandi per amministrare il sistema
- /boot : file per il boot del sistema
- /dev : file speciali che corrispondono a dispositivi
- /etc : file per la configurazione del sistema

- /home e /root : cartelle home (cartelle iniziali) per gli utenti e per root
- /lib* : librerie
- /media e /mnt : mount-point per i media removibili.
- /tmp : file temporanei
- /usr : gerarchia secondaria, quei file che possono essere montati in sola lettura

Tra pseudo terminale e processi c'è la cosiddetta disciplina di linea, quando scriviamo un programma che legge un input, non vogliamo gestire anche il fatto che l'utente cancella un carattere con backspace ecc, è un lavoro della disciplina di linea, quando un programma legge da standard input, quello che riceve è solo il risultato finale.

Cosa fa quindi la shell? In modalità interattiva, stampa un prompt (sequenza di caratteri che indica che è in attesa di comandi), legge l'input e lo spezza in token (parole e operatori), espande gli alias(stringhe che vengono espanse in qualcosa di più elevato, per esempio ls viene espando in ls-color-auto) e fa il parsing in comandi semplici e composti.
Alcuni comandi della shell vengono espansi come per esempio $var.
Le graffe possono essere usate anche per inserire delle stringhe all'interno di altre stringhe.
Usando la tilde oppure tilde + nome utente ti porta nella home directory