Bash è un linguaggio tipico dei sistemi operativi di tipo Linux.

Punto 1 - Com'è organizzato un SO?

Il Sistema Operativo (SO) è un insieme di software che si appoggia sull'hardware. Si divide in due parti principali: **la prima è il "modo kernel"**; esso si occupa di tutti gli input-output che sfruttano l'hardware per ottenere e ricavare risultati, ed ha i privilegi di amministratore; **la seconda è il "modo user"**, che è formato da tutto il software messo in disposizione da un SO che un utente può utilizzare.

Un utente, all'interno di un SO, **va confinato** - ossia i suoi permessi (e dunque il permesso dell'utilizzo di parti di hardware) vanno limitati. Ogni volta che un utente chiede a un'applicazione di realizzare un operazione, l'applicazione realizza una cosiddetta **"system call"**, ossia chiede al kernel che l'operazione venga effettivamente realizzata - se ciò è possibile l'operazione viene effettuata, altrimenti **il processo viene killato istantaneamente.**

Le system call **non sono uguali per tutti i SO**; inoltre dipendono anche dalla famiglia di CPU su cui funzionano (cambiano da Windows a Linux, ma anche da ARM a x86, ecc...). **I programmi, in un qualsiasi linguaggio, non hanno problematiche di compatibilità** grazie al fatto che il SO stesso mette a disposizione le cosiddette **"librerie di sistema"**, ossia le istruzioni base che il SO offre in base a ciò su cui esso è implementato (compatibili quindi con SO e CPU).

Nel SO sono anche presenti degli strumenti che l'utente può utilizzare per operare e interagire sul SO stesso - per esempio le shell, gli editor e i compiler/interpreti. Gli utenti sono in cima alla piramide del SO, assieme ai cosiddetti **"programmi utenti"** che utilizzano le librerie di sistema al fine di interfacciarsi con l'hardware.

Anticipazione! L'hardware non è sempre hardware (nel caso di VM), ma software che simula il comportamento dell'hardware. Inoltre la struttura appena osservata si usa anche per gestire sistemi virtualizzati.

Parte 2 - Di cosa ha bisogno il SO per funzionare?

Una CPU mette a disposizione **varie istruzioni eseguibili** - ma in base alla configurazione del "modo" della CPU alcune possono essere o non essere eseguite, a causa della loro eventuale pericolosità. I modi si chiamano "ring", e ognuno ha un numero che ne sancisce il significato:

Ring 0 - La CPU può eseguire qualsiasi istruzione macchina. Esso è utilizzabile dal kernel, che come detto prima può eseguire qualsiasi istruzione a livello hardware.
Ring 1 e 2 - La CPU esegue meno istruzioni del ring 0, ma comunque modeste a livello di pericolosità. Queste modalità vengono utilizzate dal SO stesso oppure da alcuni driver di sistema che devono operare con l'hardware.
Ring 3 - La CPU esegue solo alcune operazioni poco pericolose. E' utilizzabile dagli utenti.

Anticipazione: il kernel comanda tutto. Una VM ha comunque ha disposizione il livello 0 per se stesso: ciò crea problemi di sicurezza in quanto la VM ha più "poteri" del SO principale. Esiste pertanto un **Ring -1**, ancora più potente e privato, riservato agli Hypervisor, ossia SO che ospitano altri SO.

Parte 3 - Le System Calls

Le system calls sono di fatto delle chiamate al kernel, da parte dei programmi/utility del SO. Esse cambiano in base al SO che si usa, o al tipo di architettura della CPU. Le system calls non vengono effettuate direttamente, ma grazie a linguaggi di alto livello che, una volta interpretati da un apposito programma fornito all'interno del SO detto interprete, permettono di tradurre ciò che viene scritto da un linguaggio al linguaggio macchina, per poi eseguire le eventuali system call al kernel ed eseguire tali istruzioni. Un SO offre anche programmi di scrittura per codice di alto livello, in modo da creare i programmi stessi.

Parte 4 - L'avvio di un SO

Quando una macchina viene avviata, dalla ROM, una particolare memoria che viene preinstallata su ogni scheda madre, viene letta una serie di istruzioni macchina chiamato "bootstrap"; queste istruzioni servono per controllare se le componenti della macchina sono correttamente collegate e funzionanti, dopodiché cerca, in una lista di dispositivi di massa configurata nelle impostazioni di tale programma (modificabile in quello che un tempo si chiamava BIOS), un cosiddetto "settore di boot", ossia un settore di tale dispositivo che permette il successivo avvio del sistema operativo in esso presente.

Se è presente una configurazione MAS, il settore di boot non viene cercato solo tra i dispositivi collegati, ma anche in quelli che eventualmente non sono connessi alla macchina in modo diretto.

Parte 5 - I servizi di un SO

Un SO ha diversi servizi utilizzabili al suo interno:

- Gestione di risorse di vario tipo: dischi, processi, sicurezza e protezione di dati
- Error handling tra le periferiche della macchina e il SO
- Applicativi di input/output

- Gestione del file system, ossia dell'insieme dei file presenti in uno o più dischi
- Esecuzione di programmi con relativo error handling
- Programmi e chiamate di sistema (daemon), avviati dal primo processo avviato nel sistema operativo (in Linux è init)
- Un interfaccia a riga di comando o grafica (CLI o GUI).

L'interfaccia grafica è visualizzabile tramite un terminale, formato da monitor e tastiera; il sistema operativo, presente all'interno di un sistema, viene collegato al terminale tramite una linea TTY.
Il testo in input è detto "standard input", quello in uscita "standard output", mentre i messaggi d'errore "standard error".

Parte 6 - La CLI (Interfaccia a linea di comando) e la GUI

Una CLI è un'interfaccia che accoglie comandi in input, li interpreta e, se ritenuti contatti sintatticamente, esegue tali comandi e restituisce un output. Nelle shell moderne, vi sono delle operazioni di controllo non solo dopo la digitazione, ma anche mentre i comandi vengono digitati prima di premere "invio".

L'interprete, detto Shell, può accogliere solo due ordini:

- Comandi: sono ordini implementati direttamente nella shell, che essendo di per sé un eseguibile, possiede già delle istruzioni interne ad essa.
- Eseguibili: sono ordini non implementati nella shell, pertanto cerca se l'eseguibile citato è presente all'interno del disco (ma solo in alcune directory specificate) e, in caso positivo, esegue tale programma; in caso negativo restituisce "comando non trovato".
  Su Linux, la shell più utilizzata è bash; su Windows vi sono la Shell DOS, WSL, e PowerShell.
  Quando lavoriamo invece con interfacce grafiche (GUI), le directory diventano cartelle, le icone sono i file, e il mouse può interagire su questi due elementi con vari click diversi.

Parte 7 - Struttura del File System

Un file system è indispensabile per la gestione dei file all'interno del sistema operativo.
Lo spazio di un disco rigido è suddiviso in una o più parti, dette partizioni. Esse contengono:

- Contenitori di dati, detti files
- Contenitori di files, dette directories (o cartelle in ambiente grafico).
  In Windows, le partizioni sono chiamate tramite una lettera maiuscola. La partizione su cui Windows è installato è C.

Per navigare nelle directory, basta usare il backslash `\` per ogni directory visitata contiguamente. `C:\Windows\System32` raggiunge la cartella `System32`, all'interno della directory Windows, dentro la partizione C.

Su Linux, la partizione principale non ha un nome, ma viene identificata tramite un front slash: `/` e prende il nome di "radice" (root) del file system. Per navigare le directory si usano contiguamente simboli `/`, ad esempio `/home/vulpi/Desktop/ciao.txt`.
Per utilizzare una partizione secondaria a quella attualmente utilizzata da Linux, essa va mountata in uno specifica directory, ossia renderla effettivamente leggibile da qualche parte; per esempio, se monto una partizione su `/mnt`, allora un file "ciao.c" presente al suo interno si troverà su `/mnt/ciao.c` una volta mountata la partizione.

Non sempre i file sono accessibile da un qualsiasi utente, ma c'è bisogno di permessi di lettura o scrittura, o semplicemente accesso a una directory. Esiste un sistema di account all'interno di ogni SO, che tramite il processo di login, permette ad un utente di entrare all'interno di una cartella personale in cui ha tutti i permessi, detta home directory, dati username e password. Ogni utente è identificato univocamente dal nome e da un identificativo numerico detto User ID. In Linux, la direcotry home dell'utente vulpi è `/home/vulpi`.

- È possibile risalire alla directory attuale tramite il comando `pwd`.
- Per cambiare directory, si usa il comando `cd <directory>`; ad esempio `cd /usr/lib` ci porta ad `/usr/lib/`.
- Tramite `cd` possiamo inserire due tipologie di percorsi.
  - Percorso assoluto (l'effettiva directory): `cd /mnt/ciao.txt`
  - Percorso relativo (non parte dalla radice, ma dalla directory attuale, dove `.` rappresenta la directory attuale e `..` rappresenta la directory che contiene la directory attuale): partendo da `/home/vulpi/` e volendo arrivare a `/mnt`, si può usare `cd ../../mnt`
- Con il comando `ls <param>`, vengono visualizzati tutti i file presenti all'interno della directory. Usando il parametro `-l`, ossia "long", vengono date più informazioni su ogni file.
- Il comando `echo <args>` visualizza `args` a schermo.
  Ci sono alcune cartelle che devono necessariamente trovarsi in `/`: questo è dovuto dal fatto che alcuni file devono necessariamente trovarsi il più vicino possibile alla radice (driver, eseguibili, la bash stessa...)

Ci sono alcuni caratteri particolari, detti "metacaratteri", speciali, che svolgono funzioni se vengono digitati nella bash.

In bash è possibile dichiarare variabili, tramite una coppia simbolo-carattere.

- Per dichiarare una variabile, basta usare `<varname> = <content>`. Il contenuto può essere nullo.
- Per utilizzare, in un qualsiasi momento, una variabile, basta espanderla con `$
- Con `unset <varname>` elimina la variabile. Se la variabile viene espansa dopo aver usato `unset` essa non viene espansa.