1.5 Concetti base di Sistemi Operativi

La maggior parte dei sistemi operativi presenta alcuni concetti di base e astrazioni, quali sono **processi, spazi degli indirizzi e file**. Essi sono fondamentali per la comprensione del funzionamento del S.O.

1.5.1 I Processi

Un concetto chiave di ogni S.O è quello del **processo**.

> **DEF:** Un processo *è un programma in esecuzione* che funge anche da ***contenitore*** *che raccoglie tutte le informazioni necessarie per far girare quel programma*

Associato a ogni processo c'è il suo **spazio degli indirizzi**, *un elenco di locazioni di memoria da 0 a un massimo*, che il processo può leggere e scrivere.

Lo spazio degli indirizzi contiene **il programma eseguibile, i dati del programma e il suo stack**

In molti S.O, *tutta l'informazione riguardante ciascun processo*, diversa dai contenuti del suo spazio degli indirizzi, *è salvata in una tabella del S.O*, chiamata **tabella del processo**, **che è un array di strutture**, una per ogni processo in essere.

Un processo (sospeso) consiste quindi del suo spazio degli indirizz, generalmente chiamato **Immagine Core** e del suo elemento nella tabella di processo, *insieme al contenuto dei suoi registri e a molti altri oggetti necessari per avviare il processo successivamente*.

Le chiamate chiave del sistema di gestione del processo sono quelle che hanno a che fare con la creazione e chiusura dei processi.

> **E.g:** Un processo chiamato **interprete dei comandi** o **shell** legge i comandi a un terminale. L'utente ha appena digitato un comando con cui richiede la compilaizone di un programma. Quando quel processo ha terminato il suo lavoro, effetuerà una **chiamata di sistema** o **System Call** per terminare se stesso.

Quindi, se un processo può creare uno o più processi (chiamati ***processi figli***) e questi processi a loro volta possono creare dei processi figli (haha, process inception 😂💀), arriviamo a creare ***una struttura ad albero***. I processi relazionati che cooperano per realizzare uno specifico lavoro (E.g: Un videogioco) *necessitano di comunicare l'uno con l'altro e sincronizzare le loro attività/dati*.

Questa comunicazione è chiamata **Comunicazione tra processi** (***I**nter**P**rocess **C**ommunication*) aka **IPC**

Un processo ha a disposizione altre System Calls ***per richiedere più memoria*** *(o rilasciare memoria non più usata/necessaria),* ***attendere un processo figlio*** *che viene terminato o sovrapporre il suo programma a un altro*

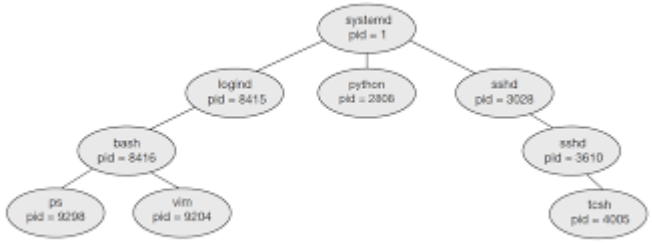


Figura 3.7 Esempio di albero dei processi di un tipico sistema Linux.

Inoltre, *ad ogni persona autorizzata* a usare il sistema è assegnato un **UID** (***U**ser **ID**entification*) dall'amministratore di sistema.

Ogni processo che parte ha l'UID della persona che lo fa partire.

Un processo figlio ha lo stesso UID del padre.

Gli utenti posso appartenere a dei gruppi, ognuno dei quali ha lo stesso **GID** (***G**roup **ID**entification*)

Un preciso UID, chiamato **Superuser** (in UNIX uwu) o **Administrator** in Windows, ha un potere speciale e può violare molte regole della protezione.

1.5.2 Spazi degli Indirizzi

Ogni computer ha una memoria principale impiegata per il mantenimento dei programmi in esecuzione.

Gli S.O più sofisticati consentono a più programmi di essere in memoria allo stesso tempo.

Per evitare che un processo interferisca con l'altro (e con l'S.O) (E.g: Il ***DeadLock***), sono stati implementati *diversi metodi*.

Anche se questi metodi di controllo *dovrebbero risiedere nell'hardware, essi vengono contrallati dall'S.O*.

Inoltre, un problema che affligeva i vecchi calcolatori era **l'uso, la gestione e la protezione della memoria.**

> **E.g:** Se un processo doveva **allocare uno spazio degli indirizzi equivalente a 2^32** (4 Gigabyte) ma il **calcolatore aveva a disposizione solo 4 Gigabyte di Memoria** (RAM), si sarebbe andati in contro a una **situazione di Deadlock**, dato che ***non sarebbe più rimasto spazio in memoria*** per gli altri processi e funzioni fondamentali del S.O

Nei nuovi calcolatori esiste una tecnica denominata **Memoria Virtuale** (in UNIX chiamata **Swap**) che essenzialmente consente al S.O di *prendere una parte dello spazio degli indirzzi in memoria* e *spostarli su un'unità di memorizzazione fisica* e ***fissa*** (E.g: SSD o HDD), cosi facendo **liberando memoria**. E quando un processo deve accedere a quello spazio di memoria, *il S.O prende quella parte di spazio degli indirizzi e la ricarica in memoria*.

Questa tecnica però ha un grave **punto dolente**, ovverò che ***in base al dispositivo di memorizzazzione*** (E.g: *Un HDD invece che un SSD*), **il processo di ricaricare quella parte degli indirizzi può richiedere un gran tempo**, portando **latenza e rallentamenti**.

1.5.3 File

Un altro concetto fondamentale di un S.O è il **File System**.

Ovvero quella parte del S.O che

> Nasconde tutte le *informazioni e dati tecnici dei dischi e dispostivi di I/O* e *li presenta al programmatore/utente* in una **forma astratta, gradevole e pulita** in modo da *facilitarne l'uso* (E.g: Cancellare, creare e modificare un documento di testo)

Ovviamente sotto la scocca **sono necessarie delle System Calls** che *eseguono le operazioni richieste*.

Per tenere traccia di tutti i file scritti e presenti sul disco, è necessario un *sistema di raggruppamento*, che in un S.O è espresso sottoforma di **Directories** (**Cartelle**)

Esempio di Cartelle e sottocartelle con file all'interno

```
Elenco del percorso delle cartelle
Numero di serie del volume: 00000099-A42E:20C9
C:\USERS\FOIS2\DESKTOP\UNI
|   .2021-11-16-Note-16-35.autosave.xopp
|   2021 B - I PROVA IN ITINERE ORE 17.pdf
|   2021-11-16-Note-16-35.pdf
|   2021-11-16-Note-16-35.xopp
|   2021-11-16-Note-16-35_v2.pdf
|   ALLEGATO_1_-__CdL_macroarea_Scienze_MM.FF.NN._2020-21.pdf
|   ALLEGATO_2_-_CdL__macroarea_Scienze_MM.FF.NN._2020-21.pdf
|   Architettura.pdf
|   AVVISO_CdL_macroarea_SCIENZE__MM.FF.NN._2020-_2021.pdf
|   Avviso_immatricolazione__corsi_laurea_accesso_libero_SCIENZE_MM.FF.NN_a.a.2021-22.docx.pdf
|   brave_5NtmOgcAPS.png
|   brave_B8Wn7DYJw4.png
|   brave_BW9kb5Fp90.png
|   brave_RZqwBqYzXJ.png
|   brave_W1a29V7SXI.png
|   domanda conferma pre-immatricolazione.pdf
|   domanda pre-immatricolazione.pdf
|   domanda.pdf
|   domanda2.pdf
|   domanda_1.pdf
|   login.pdf
|   ricevuta.pdf
|   Università.txt
|
+---Appunti Markdown
+---ARM
+---Esercizi Rossi
\---Libri Uni
        AM1 - Bramanti Pagani Salsa - 2008.pdf
        architettura-dei-calcolatori-andrew-s-tanenbaumpdf_compress.pdf
        Architettura_dei_calcolatori_Un_approccio_strutturale_by_Andrew.pdf
        C-Corso-Completo-di-Programmazione-Deitel-P.-Deitel-H.-Apogeo.pdf
        epsilon 1.pdf
        I moderni sistemi operativi.pdf
        LinguaggioC-R&K-italiano.pdf
        Matematica Discreta En.pdf
        Matematica_discreta_e_applicazioni_by_Giulia_Maria_Piacentini_Cattaneo.pdf
        MIT6_042JF10_notes.pdf
```

Ogni directory inoltre* può essere specificata* tramite il **path name** (**nome di percorso**)

Esempio di Path Name:

> `C:\Users\fois2\Desktop\Uni\Libri Uni`

> ***NOTA:*** In UNIX il simbolo per separare le diverse directory nel path name è il seguente: `/`

> Mentre su Windows viene usato il simbolo seguente: `\`

Ogni directory **è figlia della directory di partenza**, chiamata **root directory (cartella principale)**

Esempio di Root Directory e le sue Sub Directory in Windows

```
C:
    -Programmi
        - *tutte le cartelle di ogni programma installato*
            - *tutti i file di ogni programma installato*
    -Utenti
        -fois2
            -Desktop
                - *tutti i file presenti nel desktop*
            -Documenti
                - *tutti i documenti creati dall'Utente*
            - *Altre cartelle*
    -Windows
        - *tutte le cartelle dove risiedono i file essenziali per il funzionamento
del S.O*
        - *altri file essenziali*
```

Prima che un file possa essere letto, deve essere **localizzato sul disco e aperto**, **effettuare** (*se sono presenti*) **le modifiche** e alla fine occorre **chiuderlo**.

Ma prima della scrittura di eventuali modifiche, bisogna controllare i suoi **permessi**.

Se è consentito l'accesso allora il S.O restituisce un piccolo numero intero chiamato **Descrittore di File (file descriptor)** da usare nelle operazioni di modifica.

Ma **se l'accesso viene proibito**, viene restituito un **codice d'errore** e **nessuna modifica** viene apportata a quel file.

In UNIX a differenza di Windows, un **dispostivo di memorizzazione esterno**, come può essere un Floppy, CD, DVD, USB e HDD/SSD Esterni, **non viene automaticamente messo a disposizione dell'Utente/Programmatore**.

> ***NOTA:*** Nei Sistemi **Linux con una GUI**, questi dispositivi di memorizzazione esterni **vengono automaticamente messi a disposizione** dell'Utente/Programmatore, mentre nei sistemi **solo CLI**

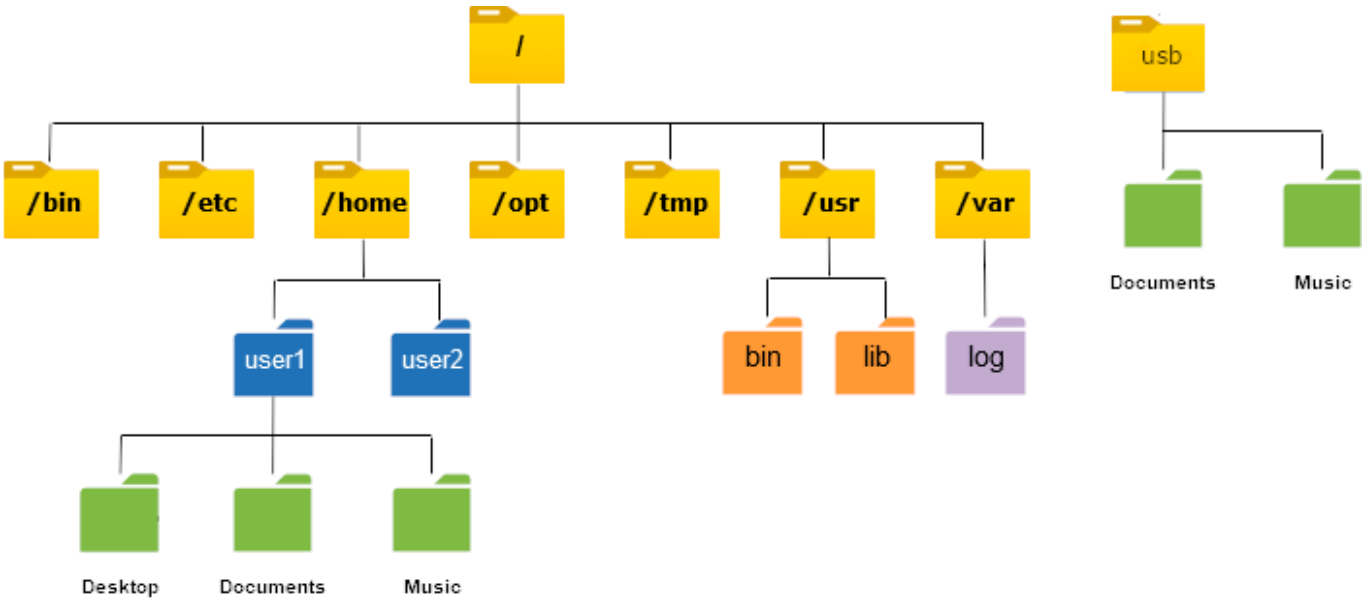
> (***C**omand **L**ine **I**nterface*) **rimane il funzionamento come sui sistemi UNIX**

Per fare in modo che questi dispotivi *vengano messi a disposizione dell'Utente/Programmatore*, bisogna prima fare il loro **Mount**.

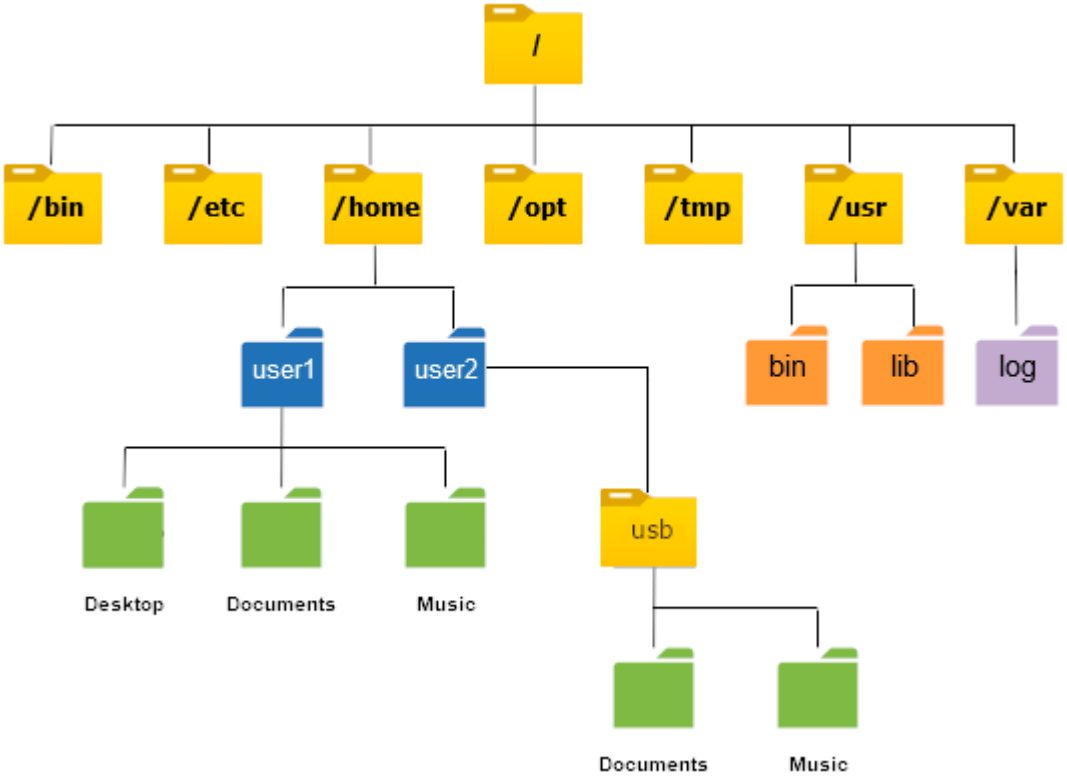
> ***DEF:*** Per **Mount** si definisce un metodo per fornire un'elegante modalità di gestione di questi supporti rimovibili.

UNIX consente al *file system presente del dispostivo removibile* **di connettersi** al *file system del sistema*.

> Esempio **prima del Mount**



> Esempio **dopo il Mount**



Un altro concetto importante in UNIX sono i **file speciali**. Questi sono derivati da una filosofia UNIX che cita:

> `Tutto è un file`

> `Everything is a file`

> ***DEF***: I file speciali sono pensati per far sì che i dispositivi di I/O siano visti come file. Questo consente che essi vengano letti e scritti con le stesse chiamate di sistema usate per leggere e scrivere i file.

Essi sono contenuti nella directory `/dev`

> ***E.g:*** Nella directory `/dev/fd0` è presente il Link Simbolico che rimanda al Floppy Drive

Un ultimo concetto riguardante sia i processi che i file sono le **pipe**

> ***DEF:*** Per **pipe** si può intendere una specie di pseudofile che può essere usato per connettere due processi.

> ***E.g:*** Esistono due processi denominati **A** e **B**. Essi posso leggere i dati dalla pipe che se fosse un file di input e vicerversa.

1.5.6 La Shell

> ***DEF:*** La **shell** è un componente fondamentale di un sistema operativo che **permette all'utente di impartire comandi** (E.g: richiedere l'avvio di altri processi) poi eseguiti dal S.O.

In UNIX esistono molte shell, fra cui *sh, csh, ksh, zsh e* ***Bash***, *la più usata* nei sistemi moderni UNIX based.

La Shell **prende comandi da dispositivi di input** (principalmente la tastiera) e **fa vedere l'output principalmente a uno schermo o a altri dispositivi**.

Una volta avviata la shell, viene mostrato un **prompt**, ovvero un *carattere simbolico* (di solito indicato da `$` nella shell Bash) che informa l'utente che **la shell è pronta a ricevere e eseguire comandi**

Eseguendo un qualsiasi comando, per esempio `date`, la **shell crea un processo figlio che eseguirà il comando** `date`, mentre il comando viene eseguito, **la shell viene messa in pausa** affinchè il **processo figlio verrà terminato**. *Appena questo accade, verrà fatto vedere a schermo il prompt e cosi via.*

Negli S.O dotati di ***GUI***, la **shell** è **disponibile** nell'applicazione grafica **Terminale.**