

# Concetti base dei Sistemi Operativi

La maggior parte dei SO presenta dei concetti base e astrazioni quali processi, spazi di indirizzamento e file.

## Processi

Un processo è fondamentalmente un programma in esecuzione a cui è associato un spazio di indirizzi.

### Spazio di indirizzi:

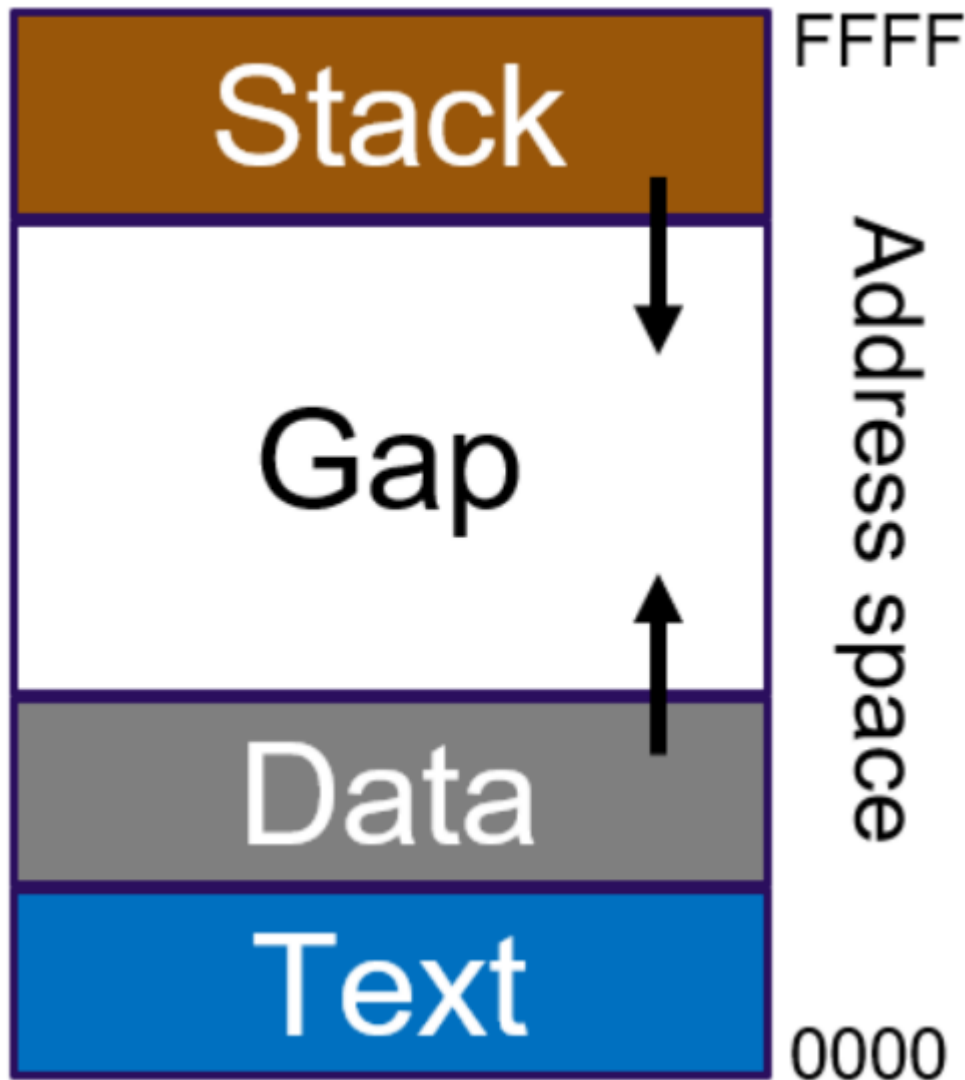
Localizzazione di memoria da 0 ad un massimo che il processo può leggere.

Questo spazio contiene: **Programma eseguibile, Dati del programma, Stack**  
Inoltre al processo sono **assegnate delle risorse** come: registri, file e allarmi.

### Il layout sopra specificato dipende da:

- 1) Architettura della macchina
- 2) Il Sistema Operativo

### 3)Programma



Possiamo pensare al processo come ad un contenitore che contiene le informazioni necessarie per eseguire il programma.

Quando un processo viene temporaneamente sospeso le informazioni riguardante il processo vengono salvate in una tabella contenente i puntatori dei file che interessano il processo sospeso.

Un processo sospeso consiste in una voce della tabella dei processi e nel suo spazio degli indirizzi.

Le principali operazioni di gestione di un processo sono:

- Creazione
- Terminazione

- **Pausa**
- **Ripresa**

Inoltre un processo può generare un processo figlio creando una gerarchia di processi sottoforma di albero.

I processi sono di proprietà dell'utente che li ha creati e questi vengono identificati tramite un **UID** che viene assegnato all'utente durante la creazione dell'account.

Inoltre i processi possono appartenere ad un gruppo di utenti identificato da un **GID**, solo gli utenti di quel gruppo possono modificare quel processo.

Uno specifico UID, chiamato **superuser o root** ha un potere speciale e può violare molte norme di protezione dei processi.

 Su UNIX, un processo figlio ha lo stesso UID del suo processo padre.

## Spazi degli indirizzi

Ogni computer ha una memoria principale che contiene i programmi in esecuzione.

Nei SO semplici ci può stare un unico programma alla volta, per eseguire un secondo programma deve essere rimosso il primo per mettere il secondo in memoria, questa operazione è detta **swapping**.

Nei SO più complessi ci possono essere più programmi in memoria, ma allo stesso tempo c'è bisogno di un meccanismo di protezione.

Esiste una tecnica chiamata **memoria virtuale** che consente di tenere parte dello spazio degli indirizzi in memoria e parte sul disco, scambiandoli secondo la necessità.

## File

Concetto fondamentale dei SO è il **file system**. Come detto precedentemente il SO deve astrarre l'interfacce dell'hardware per presentare un modello di facile uso.

La stessa cosa funziona con i file, per essere letto deve essere localizzato sul dispositivo di memoria, quindi sono fornite delle **chiamate funzionali** a questo scopo.

 I file sono astrazioni dei dispositivi di memoria

I file sono collezionati in **directory** che possono contenere:

- File
- Directory

In questo modo si crea una gerarchia chiamata **file system**

## Secondo la filosofia di UNIX: tutto è un file

La gerarchia inizia dalla **directory principale (root)**:

- /

E' possibile accedere ai file tramite **percorsi assoluti**:

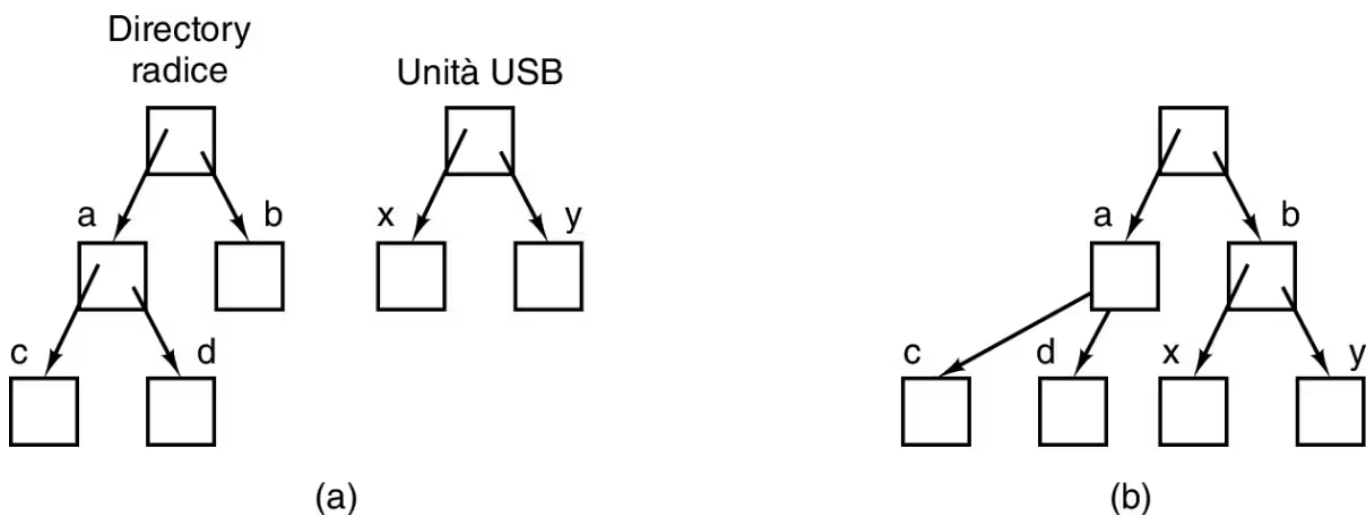
- `/home/ast/todo-list`

Oppure tramite percorsi relativi, cioè dalla directory di lavoro corrente:

- `.../course/slides1.pdf`

Altri file system possono essere **montati** nella root:

- `/mnt/windows`



In questa immagine l'unità USB viene montata e resa visibile nella gerarchia dei file.

I file sono inoltre protetti da dei **diritti di accesso**, infatti **non tutti gli utenti hanno la possibilità di lavorare con determinati file**.

Sono protetti da tuple a tre bit per:

- Il proprietario dei file
- Il gruppo
- Altri utenti.

Ogni tupla **contiene un bit per lettura, scrittura o esecuzione del file** che specifica quali utenti possono fare cosa.

### ▪ Example:

```
-rwxr-x--x myuser my group 14492 Dec 4 18:04 my file
```

- **Owner** is allowed to execute, modify, read the file
- **Group** is allowed to read and execute the file
- **Other** users are only allowed to execute the file


Altro concetto importante è quello di **file speciale**, sono pensati per far sì che **dispositivi di I/O siano visti come file** ed usano le stesse chiamate di sistema usate per leggere e scrivere file.

#### Tipi di file speciali:

- **A blocchi**: usati per modellare dispositivi costituiti da insiemi di blocchi indirizzabili in modo casuale (dischi).
- **A caratteri**: Usati per dispositivi che accettano una sequenza di caratteri in output (porte seriali o stampanti).

L'ultima cosa da tenere conto quando si parla di file sono le **pipe**.

le pipe sono **pseudo-file** che permettono la **comunicazione su un canale FIFO tra due processi**, il processo A può scrivere sulla pipe come se fosse un file di output, mentre il processo B può leggerlo come se fosse un input.

 Ricorda che la pipe va impostata in anticipo.