07-03-23

Introduzione

- I l sistema operativo gestisce le istanze dei programmi in esecuzione nel minor tempo possibile. Deve gestire una finestra temporale per la gestione della macchina in maniera ottimale.
- Un programma interrotto deve ripartire da dove si è fermato e ciò è gestito dal SO. Le informazioni che riguardano la ripartenza sono gestite dal SO e devono essere gestite nel miglior modo possibile rendendo tali operazioni "invisibili" all'utente, ottimizzando diverse sfaccettature (evitare attese fra processi ecc..).

Ci sono 2 tipi di memoria **RAM** (dati memorizzati in questo momento, quindi processi in esecuzione) e **HARD DISK**(memozzato in modo permanente)

Come viene organizzata la gestione di accesso al HD è un altro aspetto del Sistema Operativo (SO).
Il filesystem: come vengono organizzati i dati nell'HD e la lettura dell'HD è lenta e per questo il SO deve lavorare nel miglior modo possibile.
Come memorizzo i dati all'interno di un PC? In modo permanente e i dati che vengono usati in quell'istante. Un programma non ha mai una quantità di dati statica ma è dinamica perchè vengono usate strutture dinamiche e non sempre statiche. (pila, code, liste ecc..)

La memoria di un programma è sempre maggiore della dimensione disponibile in RAM e per tale motivo si usa una **memoria virtuale**.
Come vengono gestiti i processi? (esempio una stampante in un ufficio)

Segmentation fault = accedere a spazi di memoria non esistenti sulla RAM ma probabilmente sono sull'hard disk. Tale informazione va caricata nella RAM se c'è spazio altrimenti il SO deve gestire questo problema. Se la RAM è piena allora si deve rimuovere una parte di memoria per dare spazio ad altro. Se al passo dopo mi serve un dato che ho tolto al passo precedente si crea un altro **fault** e così via. Quindi i dati da rimuovere sono quelli usati meno recentemente perchè è più PROBABILE che non non verranno usati a breve.

Thread: programma che va in esecuzione suddiviso in sottoprogrammi(thread) e ognuno di essi si occupa di una specifica attività e si crea uno pseudo parallelismo.

https://www.dmi.unict.it/mpavone/so.html sito delle slide.

Sistema operativo (SO)

In caso di più processori come viene gestita la distribuzione di ciò che viene eseguito? Come si assegnano i processi ai processori? Inizialmente posso assegnarli come voglio ma devo considerare che il SO deve ottimizzare l'obiettivo da raggiungere e inoltre:

- se ho 4 processori (4 core) dovrei guadagnarci in tempistiche quindi l'obiettivo è velocizzare l'esecuzione dei programmi. *Problema*: se li assegno a caso, allora dopo un certo intervallo di tempo, o c'è un coponente che coordina i processi e gli altri 3 lavorano (coordinatore per assegnazione dei programmi ai core) -> 1 coordinatore e 3 lavorano non è l'idea migliore.
  - Se ho 2 processori che lavorano a pieno e 2 che lavorano "stanno a guardare" non mi fa raggiungere il mio obiettivo principale (cioè voglio che tutti e 4 lavorino a pieno in modo che io

possa guadagnarci)
- Nella gestione dei programmi con **più processori** ho una gestione più complicata. Ogni singolo processore deve avere **lo stesso carico di lavoro rispetto agli altri** per avere la gestione a **massimo regime**.

> I programmi in background (demoni) sono dormienti durante la loro esistenza, si risvegliano quando devono essere eseguiti e poi tornano "a dormire".

*Il **sistema operativo** mette in comunicazione i processori e altre componenti e deve soddisfare le richieste dell'utente, ovvero ricevere **immediatamente** ad un input*. (come per esempio passare da una slide ad una prossima slide).

- La **priorità** del SO è rispondere all'utente quindi tutte le altre operazioni diventano secondarie quando si deve rispondere all'utente (come su una macchina utente)
- Con una macchina industriale è diverso perchè si completa un'attività per poi passare ad un'altra quindi la priorità non è più l'utente ma varia in base all'utilizzo.

> Le operazioni di base del SO sono uguali a prescindere al tipo di SO adottato (windows, linux ecc..)

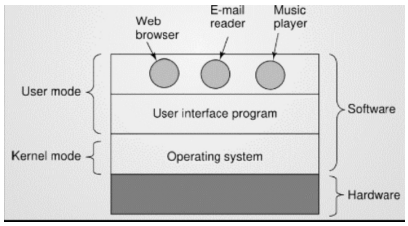
- Il SO è il *cuscinetto* fra **UTENTE** e **HARDWARE** e fa parte del software. In questo caso l'utente comunica con l'hardware mediante il linguaggio macchina.
- Il SO attiva l'hardware secondo le richieste dell'utente.

> Tutto ciò funziona grazie alla traduzione in linguaggio macchina del SO.

Modalità di suddivisione del Sistema Operativo

Lasciare che l'utente possa operare alla memoria è una scelta **azzardata** ("Salva con nome" è un'operazione che fa una chiamata di sistema. Cioè il sistema operativo prende possesso dell'incarico ed effettua l'operazione. Alla fine di ciò, si ritorna dal punto della chiamata.). Il SO è suddiviso in 2 modalità di esecuzione:

- **utente**, programmi eseguiti normalmente però con dei limiti
- **kernel**, hanno accesso a tutto senza limiti ovvero tutto ciò che è eseguito dal sistema operativo.



Ogni chiamata di sistema costa in termini di tempo (**system call**) e sono operazioni che devono minimizzare le chiamate di sistema e non togliere spazio alle operazioni dell'utente.

> *Il **cambio password** è eseguito in **modalità utente** nonostante si accedano a dati sensibili della memoria.*

Il SO gestisce le difficoltà in maniera quanto più ottimale possibili al fine di rendere ottime le prestazioni di un calcolatore in termini di velocità, memorizzazione, risposta all'utente ecc..

> *Quando si usa un PC queste operazioni sono invisibili all'utente.*

Il **compito principale** del *SO* è quello di rendere **semplice** le operazioni dell'utente e trasformarle in qualcosa di complesso (in *background*).

- Se si esegue un programma in un determinato istante non avrà lo stesso effetto se si effettua la stessa operazione in un tempo diverso perchè tutto è centralizzato in un determinato momento. (risorse disponibili in un determinato momento ecc..)
- L'ordine con cui si sceglie quale programma mandare in esecuzione cambia la totale esecuzione di un'operazione perchè dipende dall'istante di tempo.
- In caso di risorsa condivisa (stampante, file), il processo che deve accedere rimane in attesa finchè essa non si libera. Questo è un meccanismo attivo che occupa molta memoria.
- Un altro metodo è .... : se un processo non può accedere alla risorsa allora il processo si "addormenta". Quando la risorsa si libera, il processo precedente "si risveglia" e quindi il processo di attesa non è più attivo, quindi non consuma in termini di risorse e si guadagna in unità di elaborazione (hardware).
- Per il SO consumare o lasciare una casella piccolissima di memoria vuota lo valuta come uno spreco di memoria.

> *L'obiettivo comunque rimane quello di rendere bello all'utente ciò che bello non è.*