KERNEL DI UN SISTEMA OPERATIVO

mercoledì 22 marzo 2023 17:23

IL KERNEL DEL SISTEMA OPERATIVO

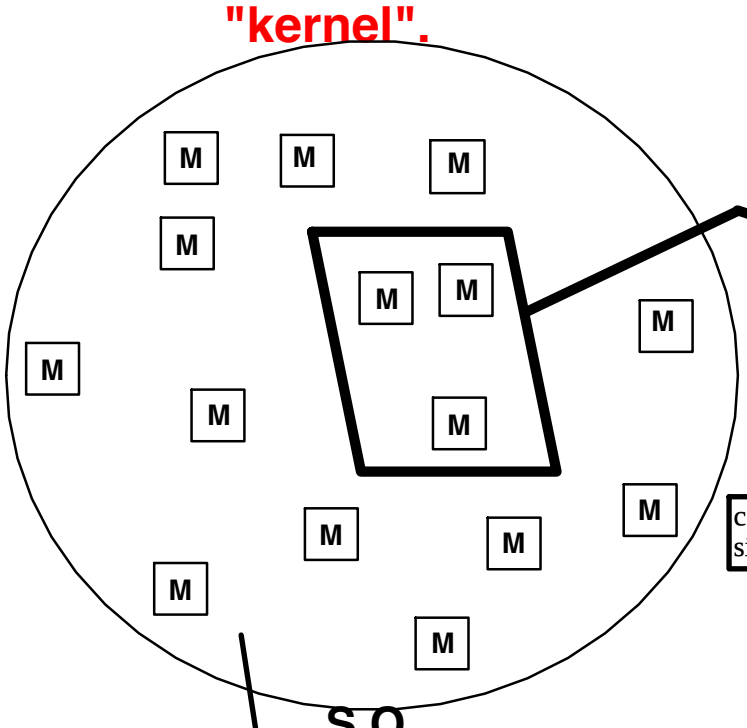
Il Kernel è un determinato gruppo centrale di moduli software del sistema operativo

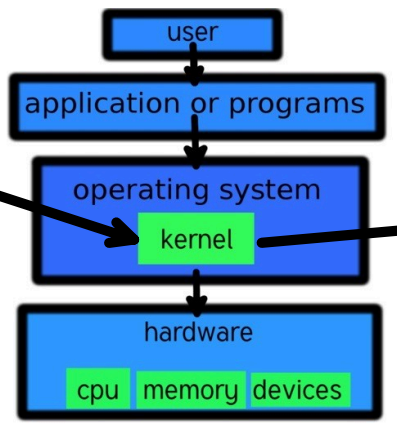
Il sistema operativo è comunque composto da moduli software.

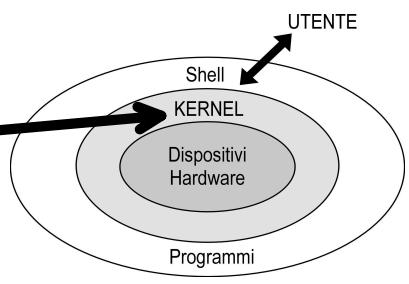
A nostra volta, all'interno di un S.O, ***possiamo separare i moduli software all'interno del S.O stesso, all'interno di gruppi****?*

Possiamo identificare un gruppo centrale all'interno del software del S.O*?*

La risposta è SI. E questo gruppo centrale si chiama "kernel".







con la denominazione kernel si intende l'insieme dei moduli software di base di un sistema operativo

Quando noi andiamo a scrivere un programma che va ad utilizzare le System Call per lavorare, utilizzando il software del S.O, quale porzione (tra Il Kernel e la parte restante) stiamo utilizzando?

Tipicamente stiamo utilizzando il Kernel.

Il kernel di un S.O è ciò che implementa i servizi utilizzabili da tutte le applicazioni utilizzando il meccanismo delle System call

Mentre invece ciò che non è Kernel, può tranquillamente non implementare questo tipo di attività.

Tutto ciò che è esterno al kernel sono programmi applicativi.
Essi sono programmi di sistema, *che tipicamente installiamo quando installiamo il sistema operativo*.

Il kernel fa anche partire alcuni di questi programmi.

I programmi che noi scriviamo sono esterni. Ma non sono programmi di Sistema Operativo.

Ma comunque vanno eventualmente ad utilizzare il kernel del s.o.

- ad esempio il kernel contiene

- i moduli accessibili tramite le system call
- i moduli per la gestione interrupt/trap

I moduli implementano attività basiche che sono necessarie ai fini dell'esecuzione di molti dei nostri programmi. E ovviamente dobbiamo avere dei moduli software che servono anche per la gestione degli interrupt.

→ **Magari non c'è più tempo per una specifica applicazione. La CPU va ceduta a qualcun altro. Chi lo fa? Il software del s.o.**

- il sistema operativo vero e proprio mette a disposizione moduli software aggiuntivi a quelli interni al kernel, come ad esempio programmi per l'interazione con gli utenti (**command interpreters**) ESEMPIO:SHELL DI UNIX
- comunemente tali programmi si appoggiano a loro volta sui moduli del kernel

- comunemente tali programmi si appoggiano a loro volta sui moduli del kernel

Qual'è lo scopo di un microkernel rispetto ad un kernel nel S.O?

Ha dei servizi cuore inferiori, in termini di numero, rispetto ai servizi cuore offerti da un'architettura basata su kernel tradizionale.

I servizi cuore all'interno di un'architettura a microkernel sono:

Gestione delle interruzioni

Non c'è verso. Se noi non siamo in grado, all'interno di una zona centrale del software, di gestire le interruzioni, non siamo in grado di gestire l'architettura hardware dove tutto il software sta andando in esercizio.

Gestione della memoria.

Se non sappiamo gestire la memoria non sappiamo attivare dei processi.

Scheduling

Ossia gestire l'assegnazione della risorsa CPU alle applicazioni che possono essere attive.

Comunicazione tra processi

Abbiamo un programma P attivo, un programma P' attivo, P può chiamare una System call per comunicare dei dati a P'. P' può chiamare una System Call per acquisire questi dati o per passare dei dati indietro a P.

Il passaggio di informazioni dal un address space ad un altro address space è un servizio del microkernel.
È importantissimo perché io posso costruire un'architettura basata su microkernel in cui quando ho un programma applicativo che ha bisogno dell'esecuzione di qualche attività, questa attività può essere implementata in un altro programma applicativo.

Le attività le codifichiamo all'esterno del kernel, invocabili dai programmi.
I programmi possono comunicare informazioni ad altri programmi affinché questi possano eseguire specifiche attività.

Allo startup il kernel viene totalmente caricato in RAM e ci rimane per tutta la durata dell'esecuzione del sistema. I programmi applicativi sono invece caricati soltanto parzialmente in memoria. Esistono system call che attivano in startup altri programmi applicativi, quindi il S.O. li carica parzialmente in RAM ed il controllo ritorna al programma appplicativo.
Se si tratta di un sistema *dual-core* due programmi possono essere eseguiti in parallelismo reale, se poi accedono entrambi a delle trap, il sistema operativo deve lavorare su più thread, quindi per rendere scalabile il sistema deve essere presente un modulo di sincronizzazione.