

SO = software che compone il HW del calcolatore

figurale fanno da | 1 o più processi
memoria centrale
dischi

le memory e i ritzi

I/O

In modo sintetico e unico

Dati di base brevi: gestiti dal software
SO.

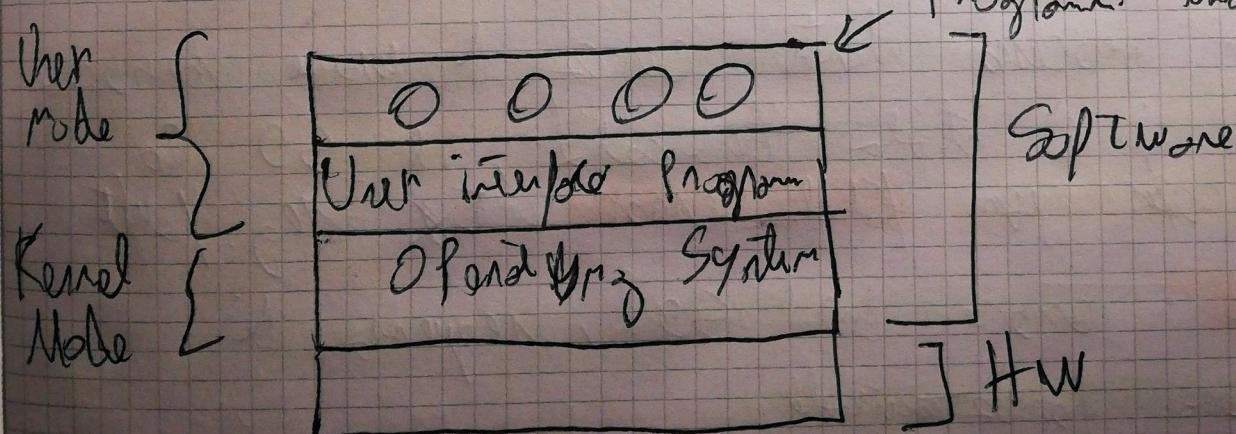
Un SO è un software che può interfacciare
direttamente con l'HW e fa fare
interfaccia tra HW e user offrendo una
serie di servizi utili con le applicazioni

Chi usa il SO? I processi

Processo: interruzione di esecuzione di un programma

User Interface: componente software che offre
interfaccie per l'uso dei servizi offerti dal SO

Programma: moduli da usare



Il software che segue è multilicato

Nobilità & no dell'HW

Lo CPU si trova direttamente in linea di ciclo.

Fetch - decode - execute.

L'operazione può già essere eseguita in linea
di nobilità. Ci sono due:

- Modello Kernel
- Modello User

Nella prima il software (quello che tutti i processi
e quei software qui lungo la call sono
vedono).

Nella seconda il software che sta nello
CPU ha delle limitazioni (molte istruzioni potranno
essere delicate).

Ogni processo ha un'area di memoria integrata
ed un suo intera le aree degli altri
processi tranne in cui si comunica tra
processi.

Quando un processo intende cercare di usare memoria
che non gli è stata assegnata, il SO lo
blocca con un **fatal error**: **Ex. Seg. fault**

KERNEL: come nel SO, requests di
istruzioni formattate.

Quando è caricato in memoria non rimane
altro che continuamente a somministrare tutto i
processi utente.

In generale ~~tutti~~ il SO è eseguito in modalità Kernel e i processi user in modalità user, però alcune componenti del SO potrebbero lavorare in modalità user. Si fa regolarmente perché queste componenti non ~~hanno~~ ~~hanno~~ mezzi per avere i privilegi quindi per eseguire le loro funzioni.

Un solo utente che esegue un SO ha concedere i minimi privilegi necessari per ogni processo.

N.B.: Utente interagisce con OS che usa il Kernel e amministra tutti i processi, mentre l'HW

Macchina virtuale

Si può vedere il SO come un'interazione b/w HW.
I servizi offerti sono strutture.

L'interazione è mai chiara per gestire le componenti, generalmente in molti semplici qualcosa di già esistente (molto complesso).

L'HW esiste, ma è complessa essere, usiamo il SO per offrire una bella interfaccia per l'uomo (questo è un'astrazione).

Esempio astrazione: FILE

Un dispositivo meccanico è composto da una sequenza di blocchi in cui possono essere scritte o lette informazioni.

Ma da un punto di vista logico, i file non esistono solo astrazioni.

Dall'entrata del File ci si può ricordare a quella del File system.

L'HW non sa cosa sia un file, se solo ricevere e leggere (in modo meccanico)

Attribuzione: Servizio offerto dal SO implementato tramite software il quale si occupa di misurare componenti software o Hardware.

SO come gestore di risorse

Un moderno SO gestisce più risorse in concorrenza fra loro

Necessita di associazione arbitrata e controllata

Risorse (via HW da SW)

La gestione è fatta con MULTIPLEXING

Il multiplexing può essere di due tipi

- SPAZIALE
- TEMPORALE

Risorse combinate temporalmente

E' la a turno per works

Il SO si occupa di gestire gestione, Ex: Stavolta

Risorse combinate spazialmente

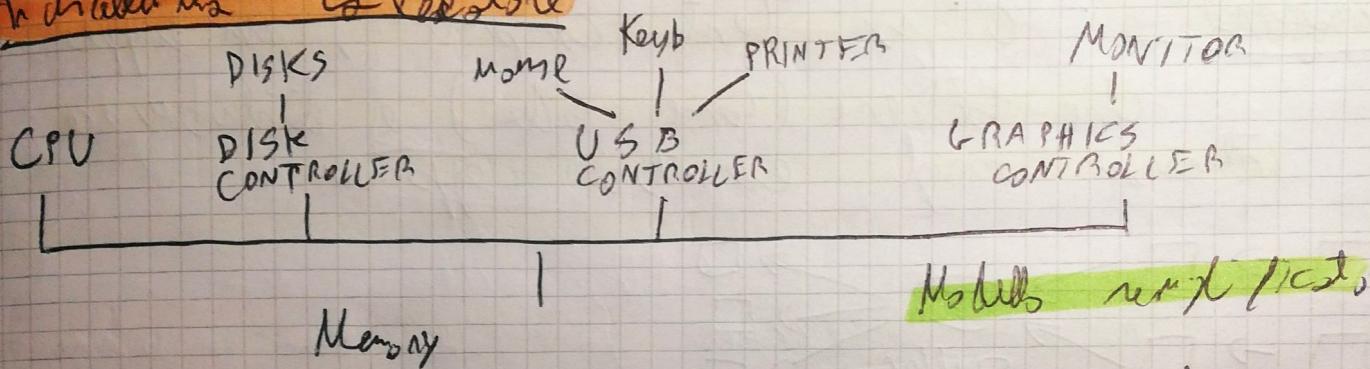
Ex: Disco fisso → Allocazione di una "zona" sul disco e gestione traccia di chi ha quale blocco

Centrale Logica CPU: dei processi fanno a turno
per poterla usare

Altro esempio esecuzione: oggi ~~CPU~~ Processore "freel" come no la CPU gli sono intrecciati pericolosamente in rete e non è GUI, oppure per la gestione di turni.

• Storia SO

• Architettura Computer



Componenti legate direttamente al bus di sistema

Processione

Basta un codice: Fetch - Decod - execute

Ecco spieghi i registi da cui ce ne sono alcuni particolari.

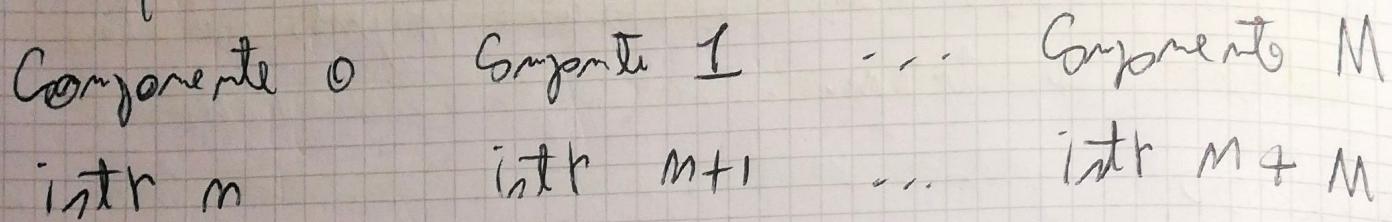
• Program Counter = ha inizio con il programma

• Stack pointer = punta allo zero dello stack, che ha il frame di ogni procedura non finita

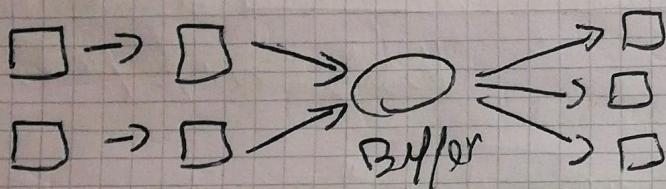
• Program Status Word = inizio di flag

Esempio per i progettatori già avviate per controllo da
sovraccarico delle istruzioni

- Pipeline: elaborazione in varie sezioni, ogni sezione
fa una cosa in una diversa istruzione



- Superpipeline: ci sono tante componenti, un buffer
che contiene (come Sba) e distribuisce
istruzioni negli esecutori opportuni

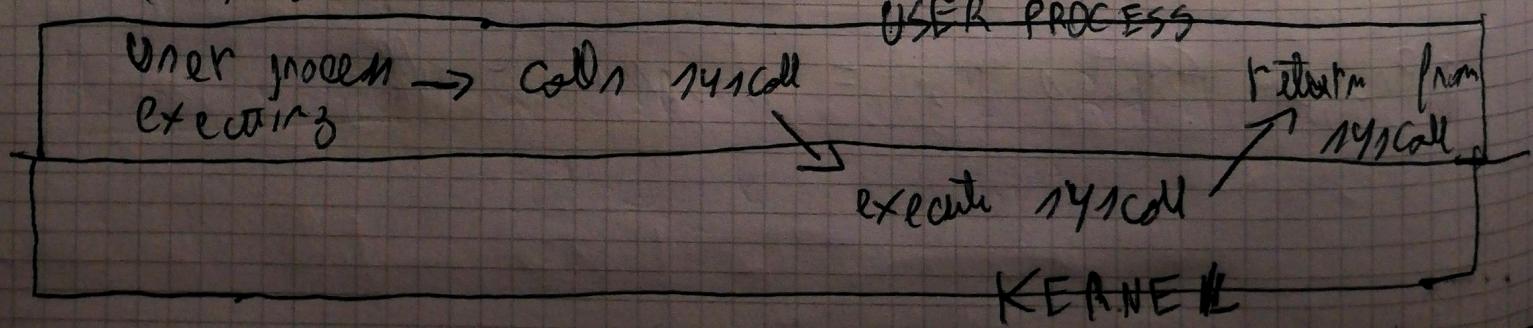


Necessità di esecuzione

I processi utente hanno bisogno di aiuto del SO
in Kernel mode in certe istruzioni, le "necessità"
sviluppate dalla chiamata a interruzione, che
entra nel Kernel e richiede il SO.

L'istruzione che switcha da User mode a Kernel
è detta TRAP.

Quando il lavoro del SO è finito si torna a
User mode



L'istruzione TRAP passa in Kernel Mode, per un motivo
a una routine nel Kernel.

Lo TRAP è una call specifica, in alcun sintesi
ha un parametro per specificare cosa deve fare

Dentro organizzazione è comunque ricca perché
"pre-programmata", la syscall indica quanto
fornire pre-programmato dal progettista del SO

Molti trap sono ottimizzati dall'HW

Ex: tentativo di divisione per 0 \rightarrow TRAP \rightarrow SO grida
in modo da interromperlo

A volte richiedono di interrompt.

Il Kernel a volte li considera per lavorare intenzionalmente.

CPU con multithreading ("Vero 2 CPU")

E se tiene all'interno della CPU 2 state di
due thread, per ottimizzazione.

Per cosa può essere utile?

La CPU a volte aspetta varie entrate informazioni
dalla memoria, invece di non fare nulla
commette null'è tra i thread e lavora su di esso
"Ammultano" il tempo morto

"E' come se avessi 2 CPU" Il SO ne tiene
comunque conto

Multiplazione; Abbiamo tanti CHIP-CPU

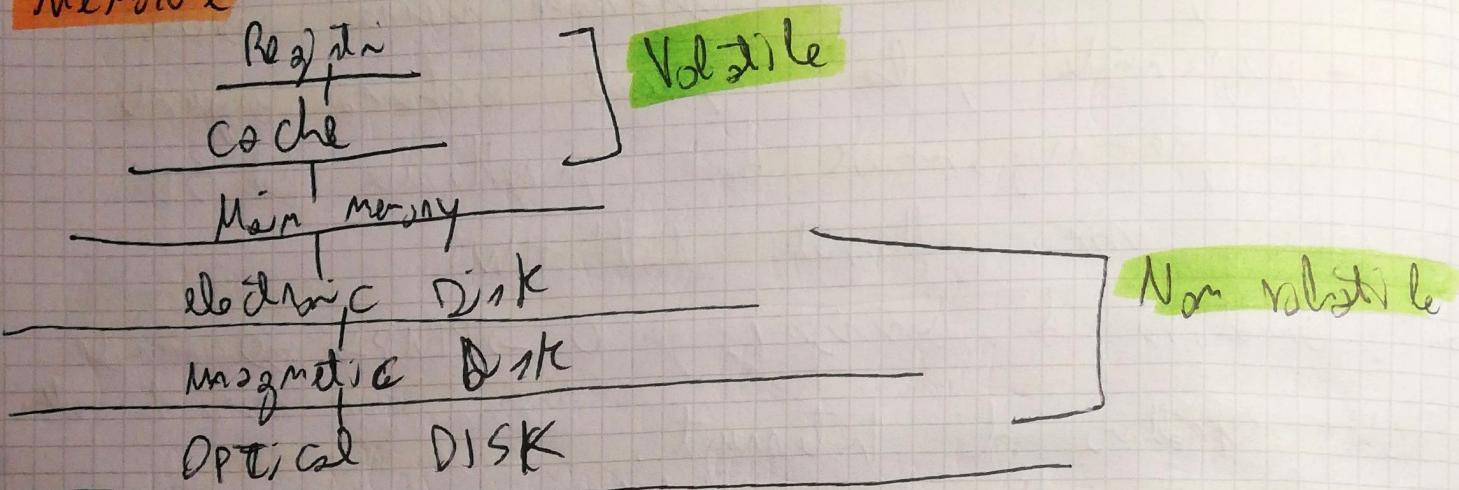
Vantaggi: più efficienza

Multi-core: la CHIP-CPU è divisa in vari CORE
ognuno organizzato in grado di fare elaborazione

Doti infinitamente tagli altri CORE

GPU: processore core migliaia di micro logic core
utili per eseguire tante piccole operazioni
Ex: 2 CPU, ognuna 2 thread \rightarrow i "rabbis" 5 CPU

Memorie



Register: veloci come la CPU, accedono non congiunta
e faticose, l'accesso va fatto dai programmi

Cache: grida molto dall'HW, i suoi cicli
sono che dicono accesso veloce.

E' organizzato a livelli

L1: all'interno della CPU, 16 KB

L2: già grande (già lenta)

Alcuni sistemi mettono una L2 grande per ogni
CPU/Core, gli altri sistemi mettono una L2 grande
compartita.

In alcuni sistemi c'è presente la L3

Main memory: generalmente alla RAM
Centinaia di megabyte

Trovano ormai sempre anche la ROM (non volatile) usata in alcuni PC per l'avvio o altro.

Diski: per mantenere dati in memoria non volatile troviamo i diski, che sono le memorie già grande e dense.

Dispositivi I/O

Infiltrano due componenti:

- Controller; gestisce interfacciamento per l'usr, invia al driver che controlla fisicamente il dispositivo.
- Dispositivo stesso; l'interfaccia è elementare ma complicata da usare

Ex: Disco SATA

C'è bisogno di un software che maneggi ogni controller e i suoi driver

Il driver interagisce col dispositivo tramite il controller, lo fa con le porte I/O

Il driver deve essere inserito nel SO, sarà così in mobilità (Kernel)

L'interazione avviene così:

- Si usa int IN/OUT ①
- Si fa una mappatura in memoria
- ① Driver eseguito in Kernel Mode, se ② riguarda

E può gestire Input Output in 3 modi diversi

1) La CPU segue il bitset che arriva I/O
e si calcola a un ciclo degli eventi in
cui si interagisce continuamente il dispositivo per
vedere se l'IO è finito.

Fatto l'I/O, se c'è bisogno il bitset mette
i biti nel punto opportuno e finisce l'esecuzione
Il metodo è detto busy waiting. Troppo noioso.

2) Usano un interruttore che sposta l'interruttore.
Il controller invia una notifica alla CPU che
mentre esegue qualcosa fare altro.
La notifica serve a regolare varie cose,
es: se ora i dati sono pronti e la CPU
può tornare a lavorare col dispositivo.

E basta usare anche quelli da i biti

~~Velocità~~ gestione degli interrupt che controlla
semplicemente la cosa degli interrupt

3) DMA: direct memory access

Già una componente hardware il chip DMA
che serve a controllare il flusso di bit tra
RAM e dispositivo rendendo meno impegnativa
la CPU.

All'inizio del processamento la CPU impone le
istruzioni sul "come lavorare"

Cataloghi SO

- Per smartphone / server:
 - Molti SO sono grandi per gestire molti processi I/O, che vengono eseguiti su server.
- Per PC; supportano multi programmazione
- Per tablet / smartphone; Android / iOS, hanno le app.
- Per sistemi integrati (embedded); installati su apparecchi, software unito, in ROM
- Real time; devono creare efficienza in gestione di eventi, non molti giri vittime in funzione. Timing fondamentale.

STRUTTURA SO

Monolitico

1) Senza supporto HW.

Non c'è separazione tra Kernel e User mode.
Da molti problemi, non è più considerato
multi programma, nessuna interattività.

2) Supporto HW per modulare User - Kernel

Chico Kernel su tutto tutto

Ogni componente può ridursi in un'altra
è uno SO gestibile

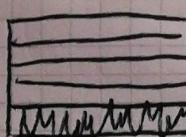
Strutture a livelli / strati

Visione verso gerarchia a livelli
 Ogni livello è controllato in base a qualità
degli suo servizi e è i più verso
offerti. Analisi in caso d' una rete di
strutture.

Strato 0 : HW

Strato 0 : SW subisce in ultima istruzione
questo struttura offre maggiore protezione

"Chi usa il servizio offerto dal livello sovridente
non dove preoccuparsi di come funziona"



SW
HW

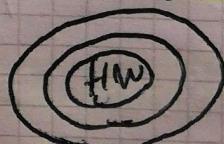
E' una buona struttura per la
continuità del servizio di rete
concrete.

Variante ad anelli concentrici (MULTICS)

La reparazione dell' HW ora è formata

Questa Variante garantisce strutture a strati anche
a tempo-time base in caso d' bug.

Mentre invece nella gerarchia livelli semplice la
riparazione è solo possibile, qui abbiamo
una struttura che è maggiormente resistente
ai wire bugs.



Questo avrà tre layer oltre molte more in avanguardia a
3 strati oltre scattered 3 richieste.

Il potere che ci riesce molte richieste a
comprare soluzioni di gestione

Microkernel

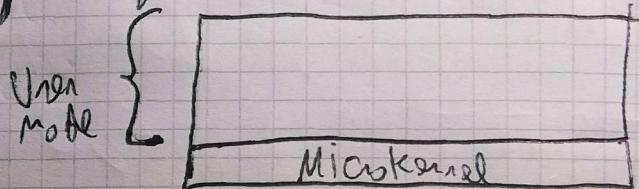
Si parla di Kernel che i occupa di scheduling, memoria e IPC. Quindi componenti che normalmente starebbero nel SO stanno "fuori". Tutto il resto è gestito da moduli o VMLinux.

La comunicazione fra servizi è fatta da messaggi, che permettono le comunicazioni anche se Kernel

Queste strutture permette un miglior design e miglior stabilità ~~dato che~~ dato che abbiano tra loro i stessi componenti.

Vantaggi: Se un modulo è buggato, molto probabilmente non renderà tamme gli altri moduli o al kernel, per questo è utile la modifica.

Strutture e moduli



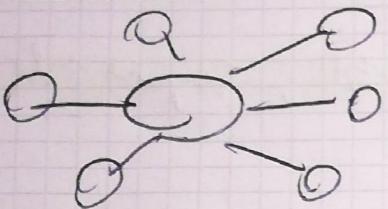
Si applica programmazione OOP al Kernel

I vari moduli implementano in modo specifico

Il kernel principale ha funzionalità già nello e i moduli sono concordi tra di sé

Abbiamo un design getto, efficiente in cui qualsiasi modulo può già invocare un altro tramite la struttura delle messaggi

La struttura ha un nucleo con "stanno" i vari moduli.



Mobilità Kernel: molto basso
nodo e tra i nodi
Nef: nesso di presa
utente

Il Kernel NON fa parte nella comunicazione
tra moduli

Macchine Virtuali

C'è un meccanismo l'hardware ha potuto alla
creazione delle macchine virtuali, **Virtualizzazione**.

Scopo: usare già SO, risparmio risorse, quelle
che fare simulazione e genere altri utenti
rimandi il concetto di macchina fisica per
le macchine virtuali come sono macchine vere,
non sono cosa dell'HW che la macchina reale
di sopra.

La macchina virtuale funziona proprio con SO

NB: La macchina virtuale in un qualche modo,
direttamente o indirettamente arriverà a usare l'HW reale.

Infatti viene a crearsi in certi casi **Overhead** fatto che deve usare la reale CPU

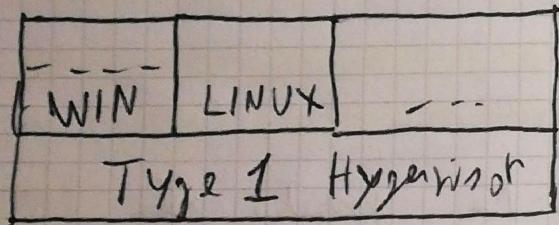
Concepto di Paravirtualizzazione: è una variante già
affidabile, il SO è adattato per funzionare sulla
macchina virtuale.

Si risparmia Over Head però ci sono già vincoli
per l'omogeneità tra macchina finca e
SO sulla virtuale.

Il tutto è gestito dall' Hypervisor , eseguendo software

Type 1

Software che gira direttamente sull' HW su cui
possono installare poi un SO (oggi)



Type 2

E' un processo che gira su un SO host in User mode.

Alcune sue componenti hanno comunque bisogno
del Kernel , per questo i richieste di presentazione
di moduli che generano ciò

Quindi :

Il software Hypervisor gira sempre in User
ma a volte deve fare cose che richiedono
Kernel Mode , il che è generato da moduli
aggiuntivi .

Oggi nei calcolatori moderni ci sono bell
supporto HW per gestire questo problema
in maniera efficiente

