# SISTEMI OPERATIVI

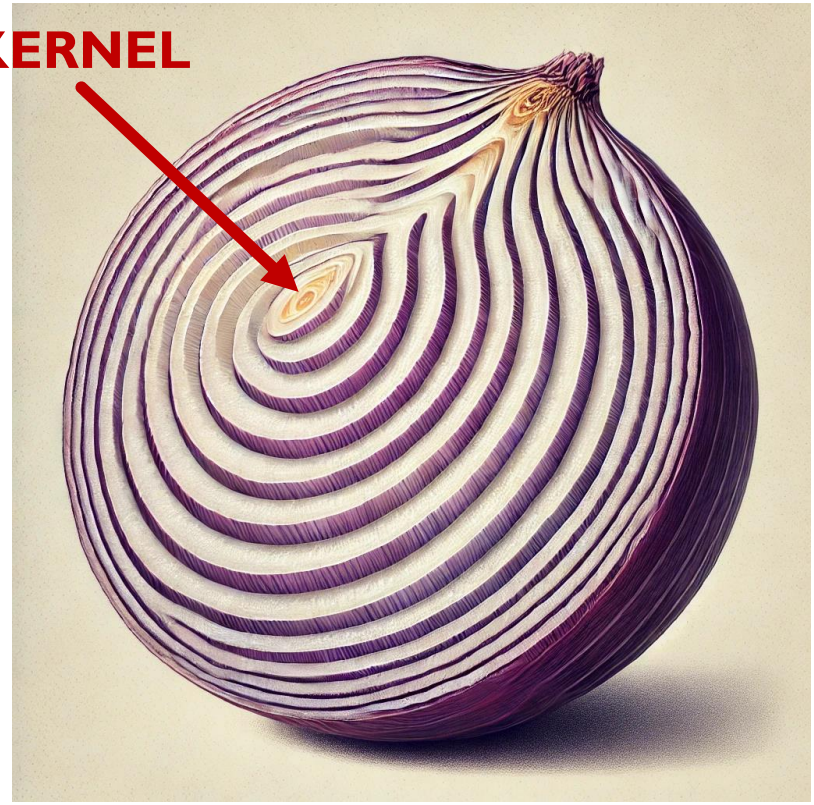
## STRUTTURA E KERNEL

# SISTEMA OPERATIVO

- Il **Sistema Operativo** è formato da un insieme di programmi organizzati tra loro in modo tale che ciascuno di essi si occupi di un compito specifico, secondo uno schema detto «**a buccia di cipolla**» (onion skin).
- I **programmi** che occupano una posizione più **interna** interagiscono maggiormente con **l'hardware**
- i **programmi** che occupano una posizione più **esterna** interagiscono maggiormente con **l'utente**.
- Il cuore/nucleo del Sistema Operativo è chiamato **KERNEL**

# SISTEMA OPERATIVO

- Il **Sistema Operativo** è formato da un insieme di programmi organizzati tra loro in modo tale che ciascuno di essi si occupi di un compito specifico, secondo uno schema detto «**a buccia di cipolla**» (onion skin).
- I **programmi** che occupano una posizione più **interna** interagiscono maggiormente con **l'hardware**
- i **programmi** che occupano una posizione più **esterna** interagiscono maggiormente con **l'utente**.
- Il cuore/nucleo del Sistema Operativo è chiamato **KERNEL**



# SISTEMA OPERATIVO (LIVELLI)

1. livello: gestore della **CPU: KERNEL**
2. livello: gestore della **MEMORIA CENTRALE**
3. livello: gestore delle **PERIFERICHE**
4. livello: **FILE SYSTEM**
5. livello: gestore dell'**INTERFACCIA** con l'utente
6. livello: **PROGRAMMI APPLICATIVI**

# SISTEMA OPERATIVO (SYSTEM CALLS)

L'insieme dei livelli definisce un **sistema centralizzato** nella quale ogni programma per poter funzionare e utilizzare risorse deve passare obbligatoriamente attraverso il **Sistema Operativo**.

Il **S.O.** quindi si frappone tra l'utente (i programmi che esso utilizza) e l'hardware sottostante.

I programmi non sono liberi di poter utilizzare le risorse hardware presenti nel dispositivo a proprio piacimento. Per effettuare una qualsiasi operazione sulle risorse è necessario chiedere al sistema operativo di effettuarla attraverso delle operazioni speciali dette **SYSTEM CALLS**

# ESEMPIO : SCRIVERE SULLO SCHERMO

1. **L'utente chiede**: il programma vuole mostrare il messaggio «Hello World!» sullo schermo. Il programma esegue quindi una richiesta (system call) al sistema operativo chiedendogli di scrivere il messaggio a video.
2. **Il sistema operativo risponde**: il sistema operativo riceve la richiesta e la gestisce utilizzando i programmi opportuni per interagire con l'hardware sottostante. Il kernel verifica se la richiesta è valida (cioè se è possibile scrivere sullo schermo) e se tutto va a buon fine, la accetta.
3. **Il messaggio viene inviato all'hardware**: dopo aver approvato la richiesta, il sistema operativo invia il messaggio al monitor attraverso l'hardware, come la scheda video.
4. **Il risultato appare**: infine, il messaggio «Hello World!» appare sullo schermo, esattamente come voleva il programma.

In questo modo, il programma non deve preoccuparsi di come funziona lo schermo o l'hardware, ma si limita a «chiedere» al sistema operativo di gestirlo.

# SISTEMA OPERATIVO

Il **Sistema Operativo** è un **PROGRAMMA**

- **Che cos'è un PROGRAMMA?** È un'entità che consuma solo risorse di memoria, in quanto risiede su disco (memoria di massa), è statico e senza vita. Sostanzialmente è un semplice **INSIEME DI BIT** (1010010101…)

# SISTEMA OPERATIVO

Il **Sistema Operativo** è un **PROGRAMMA**

- **Che cos'è un PROGRAMMA?** È un'entità che consuma solo risorse di memoria, in quanto risiede su disco (memoria di massa), è statico e senza vita. Sostanzialmente è un semplice **INSIEME DI BIT** (1010010101…)
- **PROCESSO**: È un concetto astratto utilizzato per definire un **PROGRAMMA IN ESECUZIONE**, è quindi dinamico e pieno di vita. Quando un **PROGRAMMA** da oggetto presente nella sola memoria di massa, viene trasferito in memoria centrale e inizia ad utilizzare la **CPU**, diventa dinamico e quindi **PROCESSO**. Un processo: **NASCE**, **CRESCE**, **MUORE**.

# IL KERNEL

## I° LIVELLO DEL SISTEMA OPERATIVO

# KERNEL

In un sistema/dispositivo tipicamente ci sono **MOLTI PIÙ PROCESSI DA SVOLGERE CHE PROCESSORI**.

Il S.O. riesce a fornire all'utente la sensazione che ci siano tanti processori quanti i processi in esecuzione, **VIRTUALIZZANDO LA CPU** e rendendo possibile per l'utente utilizzare più programmi «contemporaneamente».

# KERNEL

Il **KERNEL** è in grado di **VIRTUALIZZARE LA CPU** principalmente attraverso due operazioni fondamentali:

- **TIME SHARING**: ad ogni processo viene dedicato uno slot di tempo durante il quale può essere processato dalla CPU fisica. Si riescono quindi ad eseguire più processi su un unico processore.
- **CONTEXT SWITCH**: capacità del Sistema Operativo di interrompere un processo per lasciare spazio ad un altro processo, per poi riprenderlo in un secondo momento.

I processi sono completamente ignari del lavoro che compie il sistema operativo attraverso il Kernel, pensano di star utilizzando tutta la CPU per tutta la durata della loro vita. Proprio per questo motivo si parla di VIRTUALIZZAZIONE DELLA CPU poiché ogni processo pensa di utilizzare un processore fisico personale, ma in verità tutti i processi stanno condividendo lo stesso processore fisico.