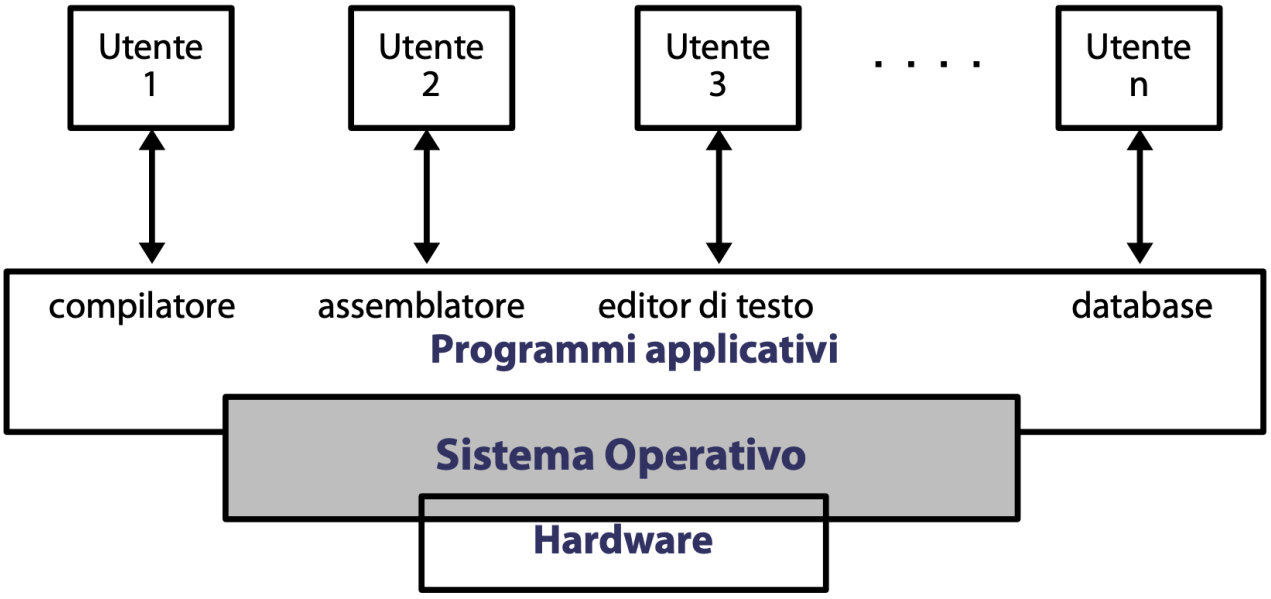
T3 - Chiamate di sistema

SO in un sistema di calcolo

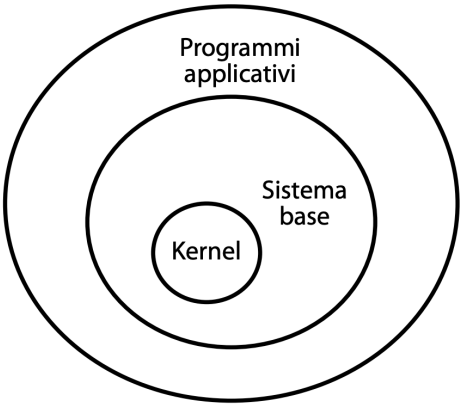


Sistema Operativo

- *Kernel*: software mediatore fra applicazioni e hw.
- *Sistema di base*: permette ad un SO di avviarsi e di presentare una interfaccia testuale all'utente.

Programmi applicativi

Tutto ciò che non è kernel o sistema di base.



Kernel

- Driver dispositivi
- Gestore I/O
- Gestore dei processi
- Gestore del file system
- Gestore della memoria
- IPC

Sistema base

- Librerie di sistema
- Caricatore dinamico
- Sistema di init
- Comandi di sistema
- Shell
- Terminale

Programmi applicativi

- Compilatori
- Interpreti
- Ambiente grafico
- Suite di ufficio
- Browser
- Client e-mail

Modello Client-Server

Una applicazione (di base e non) richiede un servizio al kernel. Il kernel elabora la risposta e la fornisce alla applicazione.

User Mode e Kernel mode

User Mode:

- Una applicazione esegue i suoi calcoli con privilegi ridotti.
- Non può alterare la memoria di altre applicazioni o del kernel.
- Non può eseguire istruzioni assembly legate all'I/O.
- Protezione contro usi maliziosi o sbadati.
  Kernel Mode:
- Quando una applicazione richiede un servizio del kernel, i suoi privilegi sono elevati al massimo rango.

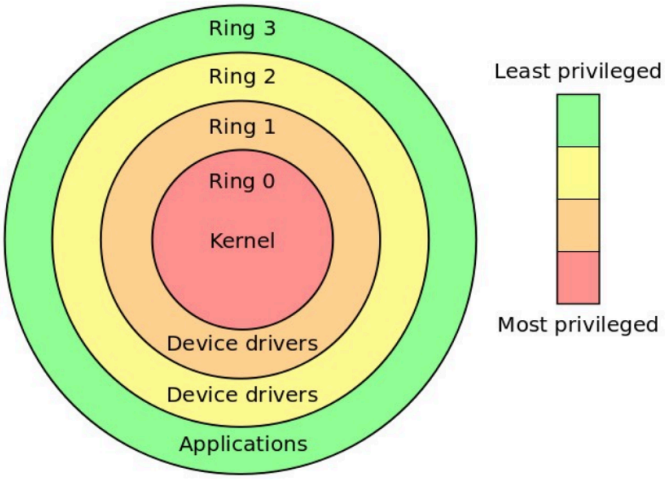
- Esegue la porzione di codice del kernel corrispettiva al servizio richiesto, al termine del servizio, ritorna in user mode.

User Space e Kernel Space

User Space:

- Insieme di tutti gli indirizzi di memoria accessibili ad una applicazione.
  Kernel Space:
- Insieme di tutti gli indirizzi di memoria accessibili al kernel.

Livelli di privilegio nelle CPU Intel



Ring 0: kernel mode.
Ring 3: user mode.

Chiamata di sistema

È l'unico meccanismo con cui una applicazione può richiedere un servizio al kernel.

- Salvataggio registri.
- Commuta user -> kernel.
- Esegue una funzione di servizio.
- Copia opzionalmente dati in memoria utente.
- Commuta kernel -> user.
- Ripristino registri ed esecuzione programma.

Passaggio di parametri

Una chiamata di sistema accetta al più sei parametri.
Se ne servono di più, occorre usare un parametro come puntatore ad una struttura dati.

Il passaggio dei parametri avviene tramite registri.
Il registro **eax** contiene sempre un identificatore numerico della chiamata di sistema.

Ingresso in Kernel Mode

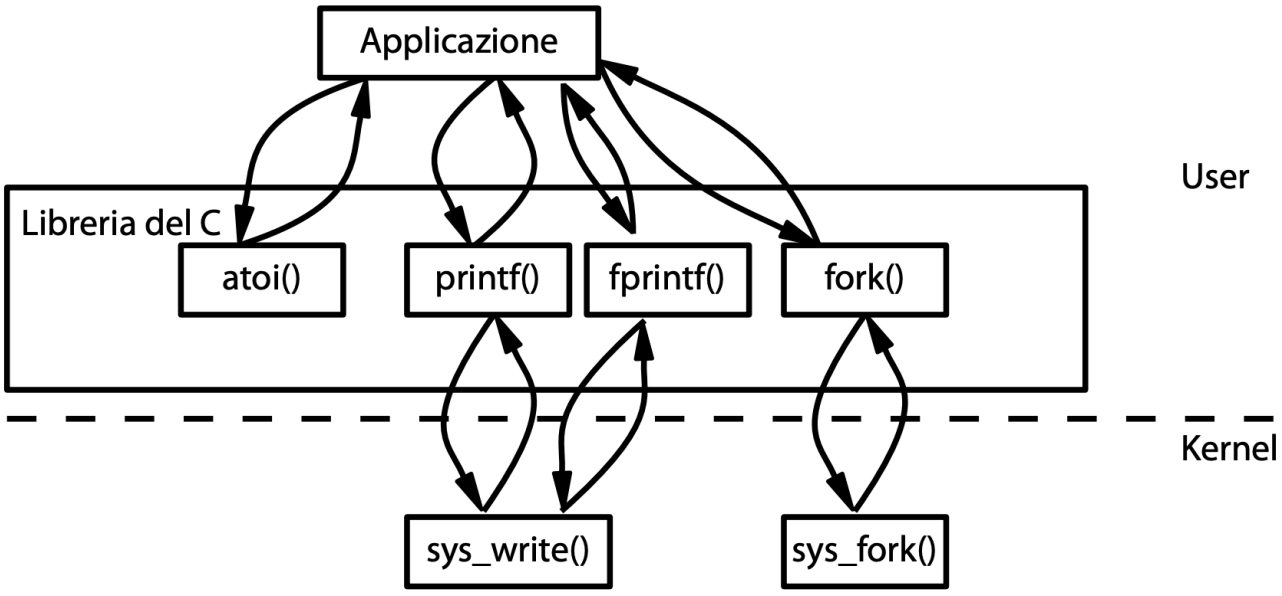
- **Fino a Pentium**: Si è usata una interruzione software (*trap*), precisamente la 128 (*int $0x80*).
- **Da Pentium 2**: Si usa l'istruzione assembly `sysenter`, ben più performante della eccezione. Ora si esegue la una funzione di servizio kernel `system_call()`. Per invocare la vera e propria funzione di servizio, che ha solitamente il prefisso `sys_`.
  Se ad esempio si invoca la chiamata di sistema `getpid()`, da qualche parte nel kernel esiste una funzione `sys_getpid()`.

Ritorno in User Space

Il valore di ritorno della funzione di lavoro è memorizzato nel registro **eax**, **rax** a 64bit.
Se si è usata l'istruzione `sysenter` per entrare in kernel mode, si usa ora l'istruzione `sysexit`.

La libreria del C

Linux usa una libreria wrapper per facilitare l'accesso ai servizi del kernel: la libreria del C (**GNU C Library**).



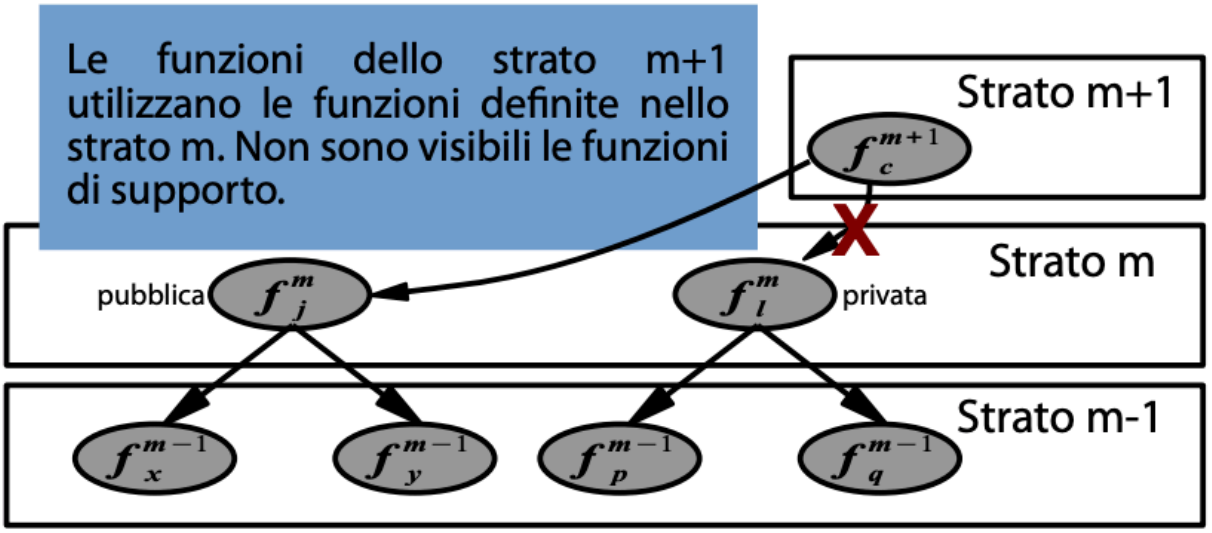
T4 - Struttura di un SO

Stratificazione

Un SO moderno è concepito per *strati* software successivi impilati uno sopra l'altro.
L'insieme degli strati forma uno *stack software*.

Uno strato contiene l'implementazione delle funzione e delle strutture dati atte a fornire la funzionalità necessaria allo strato superiore.
Implementazione in due possibili modi:

- *Software*: libreria (statica o dinamica)
- *Hardware*: chip (ROM, EEPROM)



Vantaggi

Modularità

- **Nello sviluppo**: Il primo strato può essere progettato senza alcuna considerazione per il resto del sistema. Il secondo strato si appoggia sulle funzioni del primo.
- **Nel debugging**: In un sistema ad m strati, se i primi $m-1$ sono corretti ciò implica che l'eventuale errore stia nell'$m - m_n$ strato.

Svantaggi

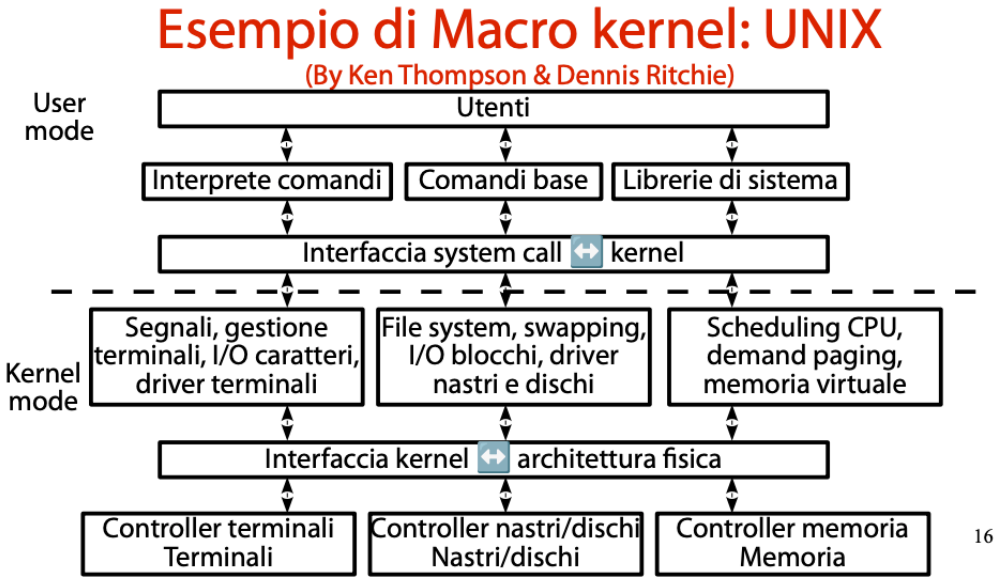
- Divisione degli strati, dove inizia e finisce uno strato?
- Riduzione dell'efficienza: ritardo nella fruizione del servizio. La funzione a strato m ci mette di meno ad essere "servita" rispetto ad una funzione nello strato $m - m_n$.

Macro, Micro, Hybrid Kernel

Macro Kernel (o monolitico)

I servizi vengono eseguiti in kernel mode. Le funzionalità essenziali del kernel sono contenute in una singola immagine eseguita al boot della macchina.
L'insieme delle applicazioni viene eseguito in user mode.

- Kernel Unix-Like (Linux, BSD, AIX, System V)
  Vantaggi:
- Esecuzione dei servizi rapida (user -> kernel / kernel -> user)
  Svantaggi:
- Forte fragilità: un crash del kernel pianta la CPU.
- Dimensione kernel enorme, possibile riduzione di prestazioni.



Dimensione della code base di UNIX

Con l'aumentare dei servizi, il kernel dei sistemi UNIX è cresciuto notevolmente.

- Problemi funzionali
- Problemi di sicurezza
- Problemi prestazionali
  Hanno portato all'uso di **moduli caricabili**, adozione dell'architettura basata su **Micro kernel**.
  Moduli caricabili: file oggetto contenente funzionalità del kernel (file system, driver di dispositivi, algoritmi di schedulazione, ...). (Dis)attivabile a tempo di esecuzione (a mano oppure automaticamente).
- Linux, FreeBSD, Mac OS X

Vantaggi

- Modularità: I vari componenti del sistema operativo, come i driver di dispositivi o le funzionalità di rete, sono separati in moduli che possono essere aggiunti o rimossi in fase di esecuzione.
- Flessibilità: I moduli possono essere caricati solo quando necessari.
- Migliore gestione delle risorse: kernel più leggero.
- Se un modulo crasha non compromette l'intero sistema.

Svantaggi

- Sicurezza: può introdurre rischi di sicurezza, poiché caricare moduli non sicuri potrebbe compromettere il sistema operativo.
- Lieve perdita di prestazione dovuta alla attivazione e disattivazione dei moduli durante l'esecuzione.

Micro Kernel

Il kernel esegue solo i servizi essenziali. Il resto è eseguito sotto forma di server applicativo.
Il kernel diventa un *sistema di messaggistica* per i server, scheduler CPU e allocatore di memoria.

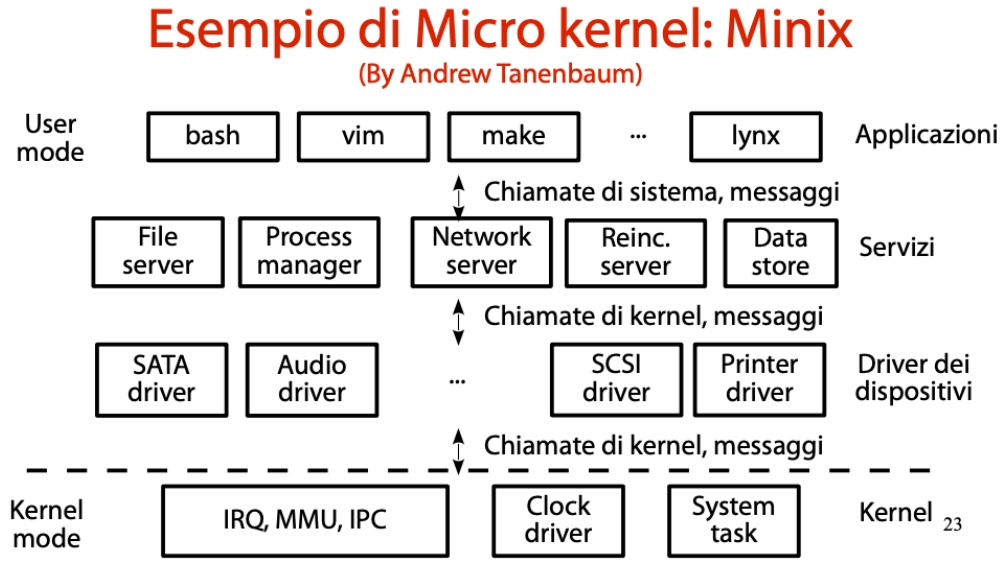
- Tru64 UNIX, Mach, Minix

Vantaggi

- Estendibili: nuovo servizio -> nuovo server, il kernel non è toccato.
- Kernel minimale, CPU cache friendly.
- SO robusto rispetto ai crash (muore il server, si ripara, se ne lancia una nuova istanza; nel frattempo, il SO continua ad eseguire).

Svantaggi

- Meno performante.
- Presenza di commutazioni user -> kernel e kernel -> user per ogni chiamata di sistema e messaggio scambiato fra server.



Hybrid Kernel

Micro Kernel, ma i driver dei dispositivi eseguono in kernel mode.
Tentativo di combinare il meglio dei Macro e dei Micro kernel. Funzionalità, prestazioni e sicurezza si collocano come intermedie fra Macro e Micro kernel.

- Windows, Plan 9, OS X