SYSTEM PROGRAMS > "Programmi di sistema"

Sono applicazioni utili a gestire il sistema.
NON sono da confondere con le system calls.
Es: Il comando "cp" per copiare

LINKING & LOADING >

Fasi finali per l'esecuzione di un programma:

LINKING

ELF in Linux ↑

Si unisce al programma compilato in file oggetto altri file oggetto dell'S.O. Questo può avvenire in maniera:

- Statica: Vengono caricate tutte le librerie dell'S.O. Diventa più pesante, ma "portable"
- Dinamica: Le librerie necessarie vengono caricate dinamicamente (DLL o Linked Objects)

LOADING

Si crea il processo, lo si associa al file eseguibile e lo si carica in memoria con uno spazio di indirizzamento definito.

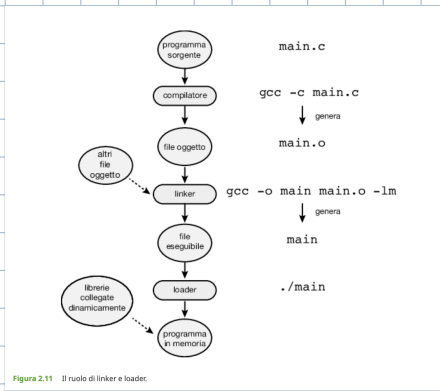


Figura 2.11 Il ruolo di linker e loader.

PROGETTAZIONE S.O. >

L'S.O. si progetta seguendo obiettivi generici dell'utente e del sistema.

SEPARATION OF CONCERNS

Nella progettazione bisogna separare:

- Policy: "che cosa fa"
- Mechanism: "come lo fa"

Questo per garantire maggiore flessibilità al S.O.

REALIZZAZIONE

I sistemi antichi erano realizzati in assembly.

Quelli moderni hanno

- core in C/C++ misto a poche parti in asm
- librerie di livello superiore in altri linguaggi di più alto livello (es: Java, Python...)

Viene scelto il C/C++ per le parti core perché consente una gestione della memoria avanzata.

STRUTTURA DELL'S.O >

Sono di diversi tipi:

- **STRUTTURA MONOLITICA**

  Tutti i servizi sono nel kernel, che viene organizzato in un singolo file con un unico spazio di indirizzamento

  - vantaggi: veloce per lateenze ridotte
  - svantaggi: Difficile da implementare ed estendere

  I kernel monolitici a loro volta possono essere

  -> **TIGHTLY COUPLED**

  Le modifiche ad un componente avranno impatto anche su altri componenti.

  -> **LOOSELY COUPLED** o "Modulare"

  Le modifiche avranno impatto solo su quel componente

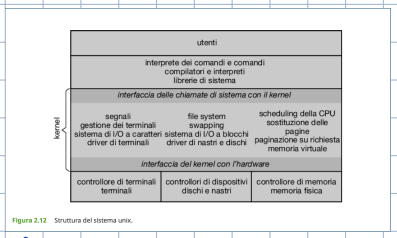


Figura 2.12 Struttura del sistema unix.

- **A STRATI**

  Si definiscono N layer con 0= HW ed N=GUI.

  Ogni strato chiama metodi degli strati precedenti

  - vantaggi: Facile progettazione
  - svantaggi: lentezza

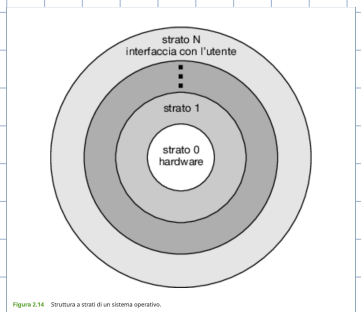


Figura 2.14 Struttura a strati di un sistema operativo.

- **Micro-Kernel**

Il kernel si occupa solo di poche attività mentre i servizi sono programmi applicativi fuori dal kernel.
In particolare il kernel:

- Comunicazione fra Processi: Tramite messaggi
- Gestione della memoria:
- Scheduling CPU

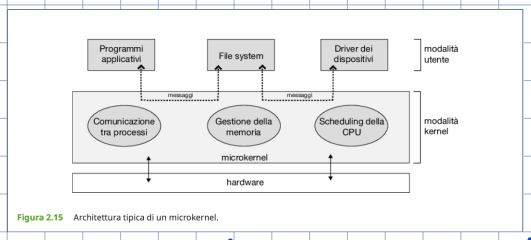


Figura 2.15 Architettura tipica di un microkernel.

Da ciò poi vi sono

- vantaggi: facile progettazione ed estensione coi moduli caricati dinamicamente
- svantaggi: Può avere latenza per la comunicazione fra processi.

- **A Moduli**: Il SO imposta le funzionalità principali nel kernel, mentre le altre funzionalità sono caricate in maniera dinamica

+ Stratificazione (come a strati) + Solo funz. principali all'inizio (micro kernel)
+ No overhead di gestione IPC

SYSTEM BOOT

Quando la macchina si accende, viene avviato il bootloader (firmware), un programma in una memoria fissa.
Esso dovrà localizzare e avviare il kernel

DEBUGGING

Solitamente si usano strumenti per analizzare i dump (file con informazioni sull'errore) che vengono creati dall'S.O.
Altre volte si può fare il tracing di parti dell'S.O. in esecuzione con debugger dedicati.

INTRO AI PROCESSI > 3.1

I S.O. possono avere 2 tipologie diverse di esecuzione dei programmi.

- **BATCH SYSTEM**

  I programmi vengono eseguiti in maniera sequenziale senza interruzioni come JOBS.

- **TIME-SHARED SYSTEM**

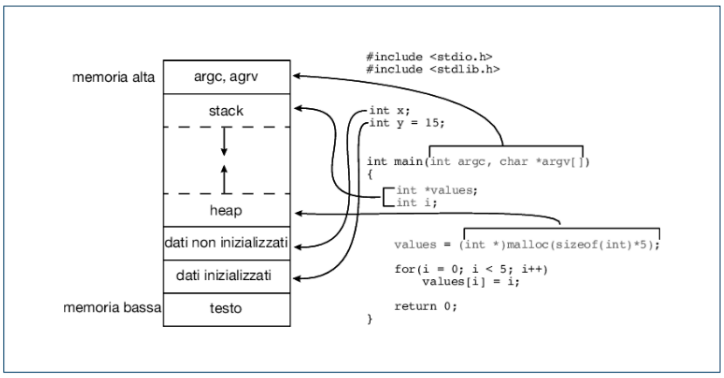
  I programmi vengono eseguiti in multitasking con i tasks.

  Ogni task è formato da
  - → Text section: Codice
  - → Registri e Program Counter:
  - → Stack: Dati e strutture statiche
  - → Data Section: SOLO per le variabili globali
  - → Heap: Dati e strutture dinamiche

STRUTTURA IN MEMORIA DI UN PROGRAMMA C

La figura seguente mostra la struttura di un programma C in memoria, evidenziando la relazione tra le diverse sezioni di un processo e un programma C reale. Questa figura è simile alla rappresentazione generale di un processo in memoria, mostrata nella Figura 3.1, con alcune differenze.

- La sezione dei dati globali è suddivisa in sezioni distinte per (a) dati inizializzati e (b) dati non inizializzati.
- È presente una sezione separata per i parametri `argc` e `argv` passati alla funzione `main()`.



Il comando gnu `size` può essere usato per determinare la dimensione (in byte) di alcune di queste sezioni. Supponendo che il nome del file eseguibile del programma C sopra riportato sia `memory`, eseguendo il comando `size memory` si ottiene il seguente output: