PROBLEMI DI PROGETTAZIONE E IMPLEMENTAZIONE PER SISTEMI DI PAGING

Danilo Croce

Dicembre 2024

GESTIONE DELLA MEMORIA: OUTLINE

- Memory Abstraction
- Virtual Memory
- Algoritmi di sostituzione delle pagine
- **Problemi di Progettazione per Sistemi di Paging**

CONSIDERAZIONI NELLA PROGETTAZIONE DI SISTEMI DI PAGINAZIONE

- La paginazione è un processo complesso che richiede una comprensione approfondita di molteplici aspetti per una progettazione efficace.
- **Altri Aspetti Cruciali**
  - **Allocazione della Memoria**
    - Scelta tra **allocazione globale VS locale** oppure **allocazione equa vs proporzionale** e come questa influisce sulla gestione delle risorse e sulle prestazioni del sistema.
  - **Gestione dei Page Fault:**
    - **Monitoraggio** della frequenza dei page fault **per ottimizzare l'uso e allocazione della memoria** e ridurre i tempi di attesa.
  - **Ottimizzazione delle Prestazioni:** Valutare le prestazioni del sistema di paginazione per massimizzare l'efficienza
    - Esempio: «*quando deve essere grande una pagina?*», «*come limitare l'uso della memoria per i processi?*»
  - **Decisioni di Progettazione:** Considerare fattori come la dimensione del set di lavoro, il comportamento dei processi, e la località dei riferimenti alla memoria per scegliere l'algoritmo più adatto.

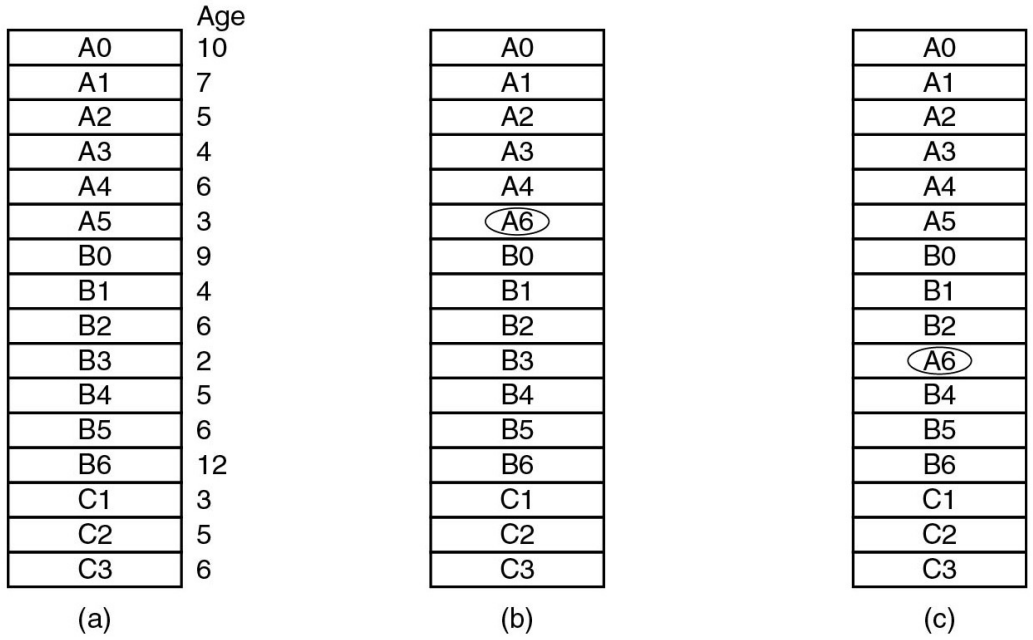
«PROBLEMI» DI PROGETTAZIONE

I PROBLEMI DI PROGETTAZIONE PIÙ COMUNI

- Allocazione Globale VS Locale
- Allocazione Equa VS Proporzionale
- Dinamica di Allocazione delle pagine
- Policy di pulizia
- Dimensione delle pagine
- Istruzioni separate e spazi dei dati
- Pagine e Librerie condivise
- File mappati in memoria

ALLOCAZIONE DI MEMORIA IN SISTEMI DI PAGINAZIONE: GLOBALE VS LOCALE

- **Allocazione Locale**
  - Ogni processo riceve una porzione fissa della memoria.
  - Semplice da implementare, ma può portare a inefficienze se il set di lavoro del processo.
- **Allocazione Globale**
  - Distribuzione dinamica della memoria tra i processi.
  - Più efficace per adattarsi alle esigenze variabili dei processi, ma richiede una gestione più complessa.
- **Esempio Pratico**
  - In Figura la differenza tra sostituzione locale (solo pagine del processo A) e globale (pagine di tutti i processi).



Il processo A ha bisogno di allocare una pagina per A6

Allocazione Locale: è possibile rimuovere solo pagine del processo A

Allocazione Globale: è possibile rimuovere pagine di qualsiasi processo

VANTAGGI DELL'ALLOCAZIONE GLOBALE DELLA MEMORIA

- **Adattabilità degli Algoritmi Globali:**
  - Gli algoritmi globali di sostituzione delle pagine si adattano meglio alle esigenze variabili dei processi.
  - Aumentano **l'efficienza quando la dimensione del set di lavoro varia nel tempo**.
- **Limiti degli Algoritmi Locali:**
  - Il **thrashing** può verificarsi con algoritmi locali **se il set di lavoro di un processo cresce oltre la memoria** allocata.
  - **La memoria può essere sprecata** quando il set di lavoro di un processo si riduce e la memoria non viene riassegnata.
- **Gestione Dinamica della Memoria:**
  - Con l'allocazione globale, il sistema operativo **deve dinamicamente assegnare e riassegnare frame ai processi**.
  - E' possibile **utilizzare i bit di aging per monitorare la frequenza di accesso delle pagine**, anche se questo potrebbe non essere sufficiente per prevenire il thrashing.
- **Sfide del Monitoraggio del Set di Lavoro:**
  - I bit di aging forniscono una **stima approssimativa**, che potrebbe non riflettere cambiamenti rapidi nel set di lavoro.
  - È fondamentale che il sistema di paginazione possa reagire in modo agile ai cambiamenti delle esigenze di memoria.

STRATEGIE DI ALLOCAZIONE DELLA MEMORIA NEI SISTEMI DI PAGINAZIONE

- **Allocazione Equa vs Proporzionale:**
  - **Allocazione Equa:**
    - Distribuzione uniforme dei frame tra i processi.
    - Esempio: `12.416` frame divisi equamente tra `10` processi risultano in `1.241` frame per processo.
    - **Svantaggi**: Non tiene conto delle diverse esigenze di memoria tra processi di dimensioni varie.
  - **Allocazione Proporzionale:**
    - Assegnazione di frame in base alla dimensione del processo.
    - Rispecchia meglio le necessità di memoria, evitando allocazioni inadeguate.
- **Importanza del Limite Minimo di Pagine:**
  - Assicurare che **ogni processo abbia abbastanza pagine** per eseguire le operazioni fondamentali.
  - **MA prevenire situazioni** in cui processi con istruzioni che **attraversano i limiti** delle pagine non possano eseguire.

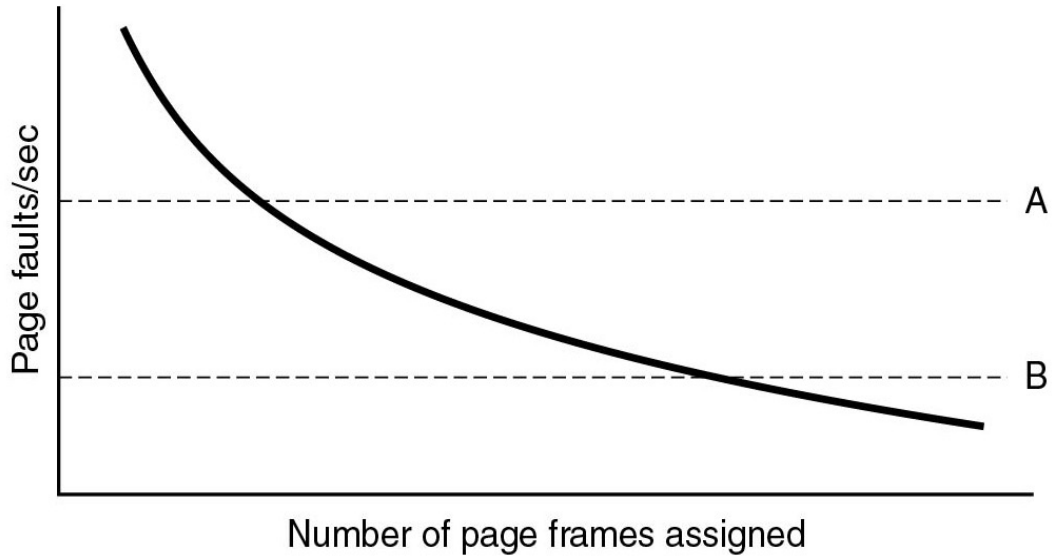
DINAMICA DI ALLOCAZIONE E ALGORITMO PAGE FAULT FREQUENCY (PFF)

- **Gestione Dinamica dei Frame:**
  - Inizio con un'allocazione proporzionale alla dimensione del processo.
  - Aggiornamento dinamico dell'allocazione in base all'evoluzione delle esigenze durante l'esecuzione.
- **Page Fault Frequency (PFF):**
  - **Monitoraggio della frequenza dei page fault per regolare l'allocazione di memoria** di un processo.
  - Aumenta i frame se i page fault sono troppo frequenti, diminuisce se sono rari.
  - Non specifica quale pagina rimuovere, focalizzandosi sulla dimensione dell'allocazione.

RELAZIONE TRA ALLOCAZIONE DI MEMORIA E PAGE FAULT

- **Relazione tra Frame Assegnati e Page Fault:**
  - Secondo algoritmi come LRU, più pagine vengono assegnate a un processo, meno frequenti saranno i page fault.
  - La frequenza di page fault diminuisce man mano che aumenta il numero di frame assegnati.

- **Monitoraggio della Frequenza dei Page Fault:**
  - Si contano i page fault per secondo e si utilizza una media mobile per tenere traccia delle fluttuazioni.
    A. **Alta frequenza di page fault indica necessità** di più frame.
    B. **Bassa frequenza** di page fault suggerisce che **il processo ha più memoria del necessario**.



GESTIONE DEL THRASHING E CONTROLLO DEL CARICO DI MEMORIA

- **Thrashing in Presenza di Allocazione Ottimale:**
  - Anche con il miglior algoritmo, **il thrashing purtroppo può sempre verificarsi** se i set di lavoro di tutti i processi eccedono la memoria disponibile.
  - Il PFF può segnalare una richiesta collettiva di più memoria senza che nessun processo possa cedere frame.
- **Strategie di Mitigazione:**
  - **Out Of Memory Killer (OOM):**
    - Processo di sistema che seleziona e termina i processi in base a un punteggio di "cattiveria" per liberare memoria.
    - Processi con elevato utilizzo di memoria o minor importanza sono tipicamente selezionati.
  - **Swapping (Scambio):**
    - Meno drastico dell'OOM Killer, sposta i processi su memoria non volatile, liberando le loro pagine per altri processi.
    - Può ridurre la richiesta di memoria senza interrompere l'esecuzione dei processi.

SCHEDULING A DUE LIVELLI E TECNICHE DI RIDUZIONE DI MEMORIA

- **Scheduling a Due Livelli**:
  - **alcuni processi sono in memoria non volatile** e solo una parte è schedulata attivamente
  - aiuta a **gestire meglio il carico** di memoria.
  - utile per ridurre occupazione di memoria di **processi in background in sistemi interattivi**
- **Gestione della Multiprogrammazione**:
  - La selezione dei processi da spostare considera anche caratteristiche:
    - Sono processi CPU bound o I/O bound
    - Qual è la dimensione e/o frequenza di paginazione dei processi
- **Altre Tecniche**:
  - Oltre a «uccidere» o spostare processi, si possono usare **compattamento**, **compressione** e **deduplicazione** (same page merging).

POLICY DI PULIZIA E PAGING DAEMON

- **Contesto**: La policy di **pulizia è un aspetto critico** nella gestione della memoria.
- **Aging e Frame Liberi**: L'aging è più efficace con molti frame di pagina liberi disponibili.
  - Se i frame sono tutti occupati e modificati, occorre scrivere le vecchie pagine in memoria non volatile prima di caricarne di nuove.
  - È **preferibile mantenere un buon numero di frame di pagina liberi** piuttosto che occupare tutta la memoria e cercare frame liberi solo al bisogno.
- **Paging Daemon**: Un processo in background usato dai sistemi di paginazione
  - **Inattivo per la maggior parte del tempo**, si attiva periodicamente per controllare lo stato della memoria.
  - Quando i frame liberi scarseggiano, inizia a selezionare pagine da rimpiazzare utilizzando un algoritmo di sostituzione delle pagine.

POLICY DI PULIZIA E PAGING DAEMON (2)

- **Scrittura in Memoria Non Volatile**: Se le pagine sono state modificate, vengono scritte in memoria non volatile.
  - I contenuti precedenti delle pagine vengono conservati, permettendo un eventuale rapido ripristino.
- **Implementazione con «Clock a Due Lancette»**
  - La **lancetta anteriore** (gestita dal paging daemon) **avanza scrivendo le pagine sporche in memoria non volatile** e procede senza azioni ulteriori sulle pagine pulite.
  - La lancetta posteriore si occupa della sostituzione delle pagine
    - maggiore probabilità di trovare pagine pulite (grazie al lavoro del paging daemon).

DIMENSIONE DELLE PAGINE E BILANCIO DEI FATTORI

- **Selezione Dimensione Pagine**: I sistemi operativi possono selezionare la dimensione delle pagine
  - Esempio: unendo due pagine da 4096 byte per formarne una da 8 KB.
- **Fattori a Favore di Pagine Piccole**: Riducono la frammentazione interna (spazio sprecato nelle pagine parzialmente vuote) e l'utilizzo di memoria
  - un programma potrebbe richiedere meno memoria con pagine più piccole.
- **Svantaggi delle Pagine Piccole**: Richiedono **tabelle delle pagine più grandi** (più voci) e possono aumentare il tempo e lo spazio necessario per il trasferimento di dati e la gestione della memoria.

DIMENSIONE OTTIMALE DELLE PAGINE E THP

- **Dimensione Ottimale**: Determinata equilibrando frammentazione interna (favorevole a pagine più grandi) e overhead della tabella delle pagine (favorevole a pagine più piccole).
  - Vedi slide successiva
- **Pagine di Diverse Dimensioni**: Alcuni sistemi operativi utilizzano pagine di diverse dimensioni per parti diverse del sistema (ad es., pagine grandi per il kernel).
- **Transparent Huge Pages (THP)**: Tecnica per utilizzare pagine di grandi dimensioni ottimizzando l'uso della memoria, spostando la memoria del processo per creare intervalli contigui.

CALCOLO DELLA DIMENSIONE OTTIMALE DELLE PAGINE

- **Parametri Considerati**:
  - Dimensione media del processo: s byte (esempio 1MB).
  - Dimensione della pagina: p byte (da calcolare).
  - Dimensione di ogni voce nella tabella delle pagine: e byte (esempio 4 o 8 byte).
- **Calcolo Overhead**:
  - Numero di pagine per processo: $\approx s/p$.
  - Spazio occupato nella tabella delle pagine: $s \cdot e / p$ byte.
  - Memoria sprecata per frammentazione interna nell'ultima pagina: $p/2$.
    - **Fenomeno dell'Ultima Pagina:** Per qualsiasi processo, l'ultima pagina di memoria allocata potrebbe non essere completamente riempita. Esempio: un processo richiede 10.5 KB di memoria la pagina è di 4 KB, il sistema dovrà allocare 3 pagine lasciando 1.5 KB di spazio inutilizzato nell'ultima pagina.
- **Overhead totale**: $se/p + p/2$:
  - Il primo termine (tabella delle pagine) aumenta con pagine più piccole.
  - Il secondo termine (frammentazione interna) aumenta con pagine più grandi.
  - L'**ottimo si trova bilanciando questi due fattori**.

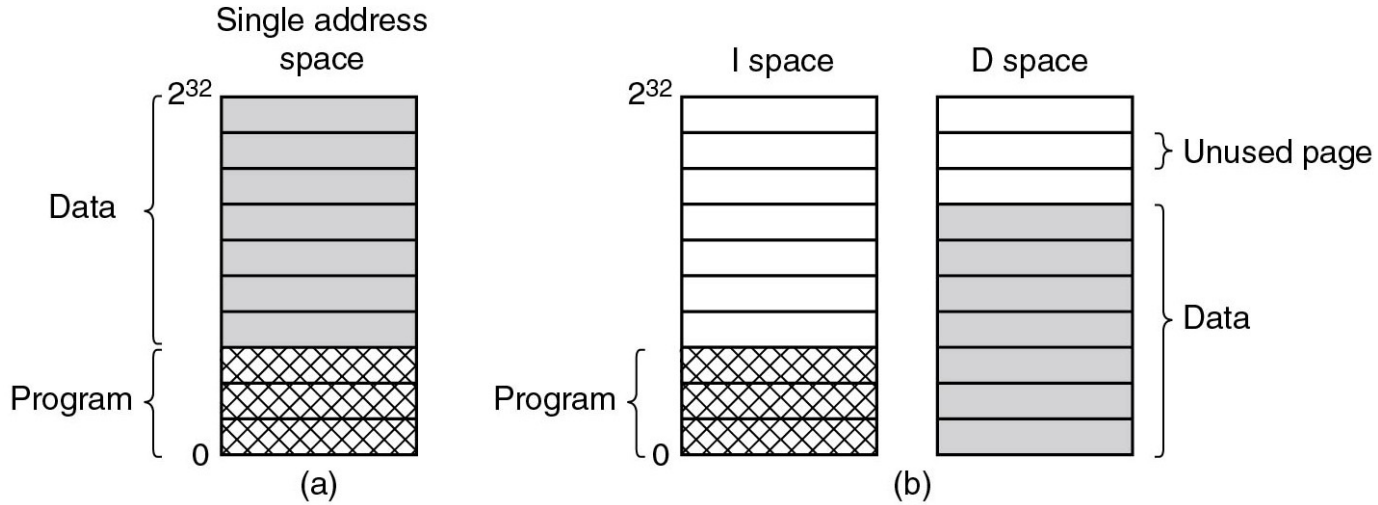
CALCOLO DELLA DIMENSIONE OTTIMALE DELLE PAGINE (2)

Overhead totale: $se/p + p/2$

- **Formula per la Dimensione Ottimale delle Pagine**:
  - Derivata della funzione di overhead rispetto a p uguagliata a zero: $-se/p^2 + ½ = 0$.
  - Dimensione ottimale delle pagine: $p=\sqrt{2se}$.
  - Esempio: Per s = 1 MB e e = 8 byte, p ottimale è 4 KB.
- **Variazione nelle Dimensioni delle Pagine**:
  - Gamma tipica in computer commerciali: da 512 byte a 64 KB.
  - Dimensione comune attuale: 4 KB

PROBLEMI DI PROGETTAZIONE: SPAZI SEPARATI PER ISTRUZIONI E DATI

- La maggior parte dei computer ha un unico spazio di indirizzamento condiviso da programma e dati. In passato alcuni sistemi avevano uno spazio di indirizzamento separato per istruzioni e dati.



- Al giorno d'oggi si vedono ancora spazi Istruzioni e Dati separati nelle cache, nei TLB, Cache L1
  - Dove lo spazio è poco, si tende a separare istruzioni (più importanti) dai dati.

CONDIVISIONE DELLE PAGINE NEI SISTEMI MULTIPROGRAMMATI

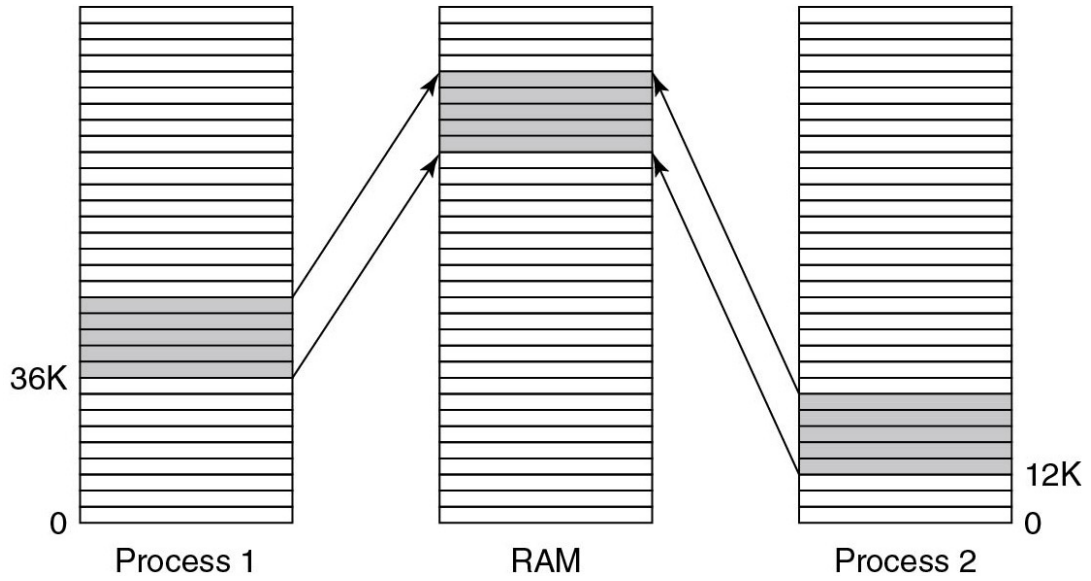
- **Motivazione della Condivisione**
  - E' comune che **molti utenti eseguano lo stesso programma** o **utilizzino le stesse librerie**.
  - **Condividere pagine di memoria** tra questi processi è più **efficiente** che mantenerne copie separate.
- **Tipi di Pagine Condivisibili**
  - Le **pagine di sola lettura**, come il testo dei programmi: **SI**
  - Le pagine dei dati: generalmente **NO**
- Per facilitare la condivisione è meglio separare spazi di indirizzo in:
  - **I-space**: Istruzioni
  - **D-space**: Dati
- **Processi diversi** possono utilizzare la **stessa tabella delle pagine per l'I-space** ma tabelle diverse per il D-space.
  - **Implementazione e Scheduling**: con ciascun processo ha puntatori sia all'I-space che al D-space.
  - Lo scheduler utilizza questi puntatori per impostare l'MMU.

NON E' TUTTO ORO: GESTIONE DELLE PAGINE CONDIVISE E COPY ON WRITE

- **Problemi con Pagine Condivise**
  - La **rimozione di un processo** da memoria può causare **numerosi page fault in un altro processo** che condivide le stesse pagine.
  - È cruciale sapere se le pagine sono ancora in uso per evitare la loro liberazione accidentale.
- **Condivisione dei Dati**: Più complessa rispetto alla condivisione del codice!
  - Ad esempio, in UNIX, dopo una `fork`, genitore e figlio condividono sia il testo che i dati, inizialmente come sola lettura.
- **Copy on Write (Copia in Caso di Scrittura)**: Se un processo modifica i dati, si genera una trap, e viene creata una copia della pagina modificata.
  - Entrambe le copie diventano poi modificabili (READ/WRITE).
  - **Questo metodo evita la copia di pagine che non vengono mai modificate.**
  - Estremamente efficiente per evitare la proliferazione di pagine

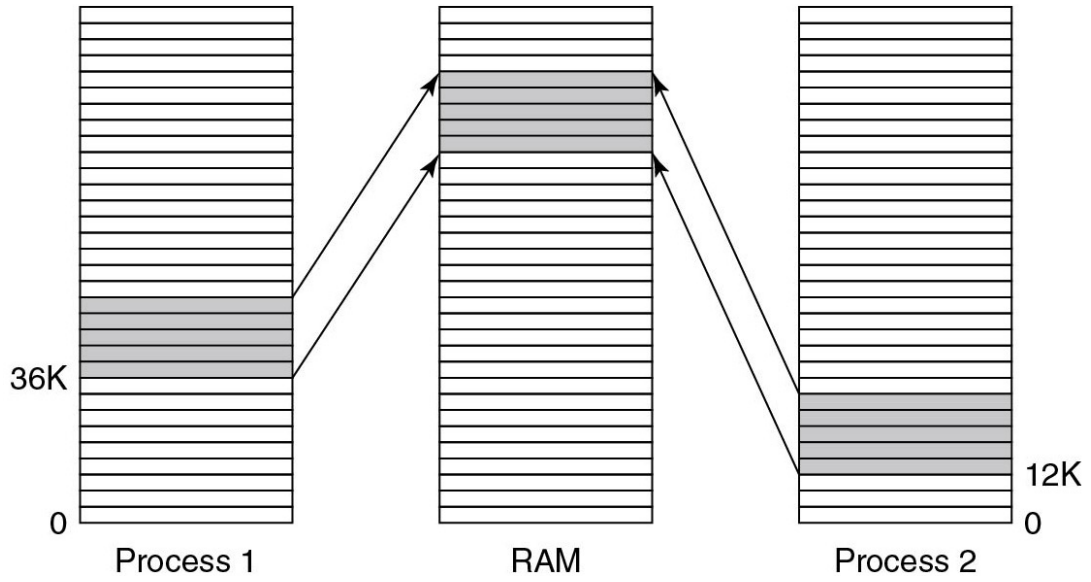
LIBRERIE CONDIVISE - PRINCIPI E FUNZIONAMENTO

- **Condivisione su Ampia Scala**:
  - I SO condividono automaticamente tutte le pagine di testo di un programma avviato più volte.
  - Per evitare problemi, meglio pagine in sola lettura
- **Copy on Write per Dati**: Se un processo modifica una pagina di dati condivisa, occorre applicare "*copy on write*".
- **Librerie condivise - Dynamic Link Libraries (DLLs)**: Usate per ridurre l'ingombro di grandi librerie comuni.
  - **Vantaggi**: Risparmio di spazio



LIBRERIE CONDIVISE - PRINCIPI E FUNZIONAMENTO

- **Problema di Indirizzamento:** Le librerie condivise possono essere posizionate a indirizzi diversi nei vari processi.
  - Questo **impedisce l'uso di indirizzi assoluti** nelle istruzioni.
- **Soluzione Compilativa:** Le librerie condivise vengono **compilate con indirizzi relativi** anziché assoluti.
  - le istruzioni usano offset relativi piuttosto che puntare a indirizzi specifici.



FILE «MAPPATI» IN MEMORIA E IL LORO IMPIEGO

- **Concetto**: I file mappati consentono a un processo di mappare un file all'interno del proprio spazio di indirizzi virtuali.
  - **Funzionamento**: Alla mappatura, **nessuna pagina viene caricata immediatamente**.
    - Sono paginate su richiesta dalla memoria non volatile, man mano che vengono "toccate"
  - **Scrittura su File**: Quando il **processo termina** o la mappatura è eliminata, **tutte le pagine** modificate vengono **riscritte sul file**.
- **Modello I/O Alternativo**: Offre un modo diverso di eseguire I/O, permettendo di accedere al file come se fosse un grande array di caratteri in memoria.
- **Comunicazione tra Processi**: Se più processi mappano lo stesso file contemporaneamente, possono comunicare attraverso questa memoria condivisa.
  - Le modifiche apportate da un processo sono immediatamente visibili agli altri.