Lezione 19 [25/11/22]

Capitolo 9: Memoria Virtuale

- Introduzione
- Paginazione su richiesta
- Copiatura su scrittura
- Sostituzione delle pagine
- Allocazione dei frame
- Attività di paginazione degenere (trashing)
- File mappati in memoria
- Allocazione di memoria del kernel
- Altre considerazioni
- Esempi tra i sistemi operativi

Introduzione

La memoria virtuale è una tecnica che permette di eseguire processi che possono anche non risiedere completamente in memoria

Permette la separazione della memoria logica dell’utente dalla memoria fisica:

- Solo parte del programma viene caricato in memoria per essere eseguito
- Lo spazio di indirizzi logici può quindi essere più grande dello spazio degli indirizzi fisici

- Permette di condividere lo stesso spazio di indirizzi tra più processi
- Permette tecniche più efficienti di creazione di processi

Parliamo quindi di **spazio degli indirizzi virtuali**: è la collocazione dei processi in memoria dal punto di vista logico (o virtuale)

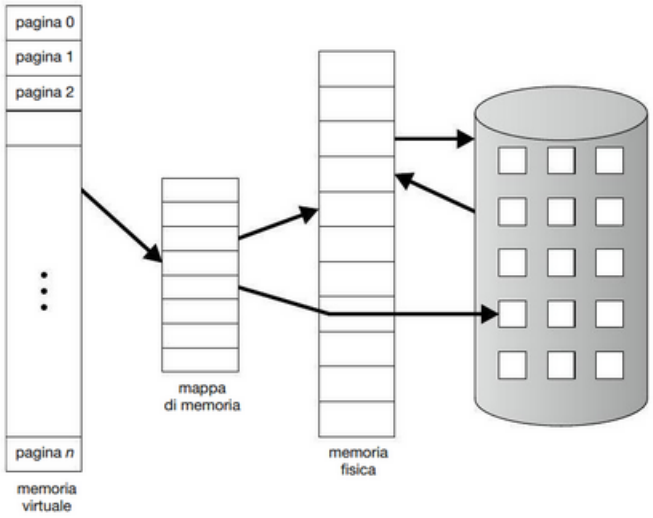
La scelta della memoria virtuale permette all'heap di crescere verso "l'alto", poiché esso ospita la memoria allocata dinamicamente, e allo stack di crescere verso "il basso". Lo spazio che si trova tra lo heap e lo stack è parte dello spazio degli indirizzi virtuali, ma richiede pagine reali solo se heap o stack crescono

Uno spazio di indirizzi con dei buchi è definito sparso: è utile nel caso di espansione di heap e stack, o per il collegamento di librerie nei buchi

La memoria virtuale può essere implementata tramite:

- **Paginazione su richiesta**
- **Segmentazione su richiesta**

Schema di memoria virtuale più grande della memoria fisica:



Paginazione su richiesta

La maggior parte dei sistemi operativi utilizza il metodo della paginazione su richiesta

Anziché caricare in memoria l'intero processo, come nella paginazione, si carica una pagina in memoria solo quando essa è necessaria

- Per caricare un programma utente in memoria è necessario un minore tempo di I/O (aumentando la velocità di esecuzione)
- Nessun vincolo sulla quantità di memoria fisica necessaria ad un processo. Gli utenti possono scrivere in uno **spazio degli indirizzi virtuali** molto grande
- Aumento del grado di multiprogrammazione grazie al minore utilizzo della memoria fisica

La memoria virtuale si fonda sulla separazione della memoria logica dalla memoria fisica

Una pagina è necessaria → c'è stato un riferimento ad essa

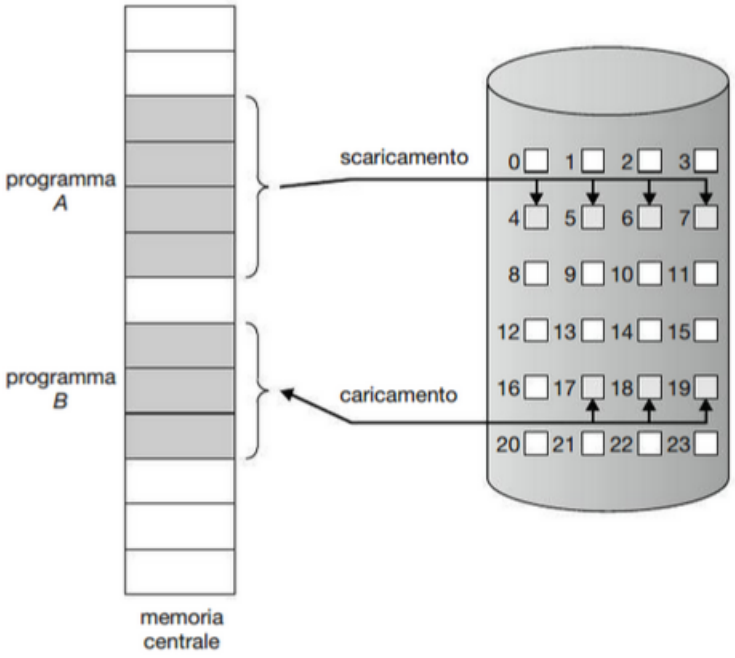
Per eseguire un processo occorre caricarlo in memoria. Per fare ciò si utilizza un metodo di avvicendamento “pigro” (**lazy swapping**): non si carica mai in memoria una pagina che non è necessaria

In questo ambito il termine swapping non è appropriato: è un **paginatore** (**pager**) a gestire le singole pagine dei processi
Esso fa ipotesi sulle prossime pagine che verranno utilizzate prima che il processo venga scaricato

Se il processo tenta di accedere ad una pagina che non era stata caricata nella memoria, il sistema genera una **eccezione di pagina mancante (page fault trap)**

- Il riferimento non era valido → si termina il processo (abort)
- Il riferimento era valido → si porta la pagina in memoria

Schema di memoria virtuale più grande della memoria fisica:



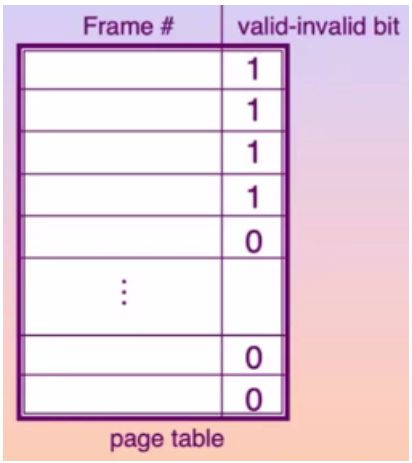
Bit di validità

A ciascuna pagina è associato un bit di validità:

- **1: in memoria**
- **0: non in memoria oppure non appartiene allo spazio di indirizzi logici del processo**

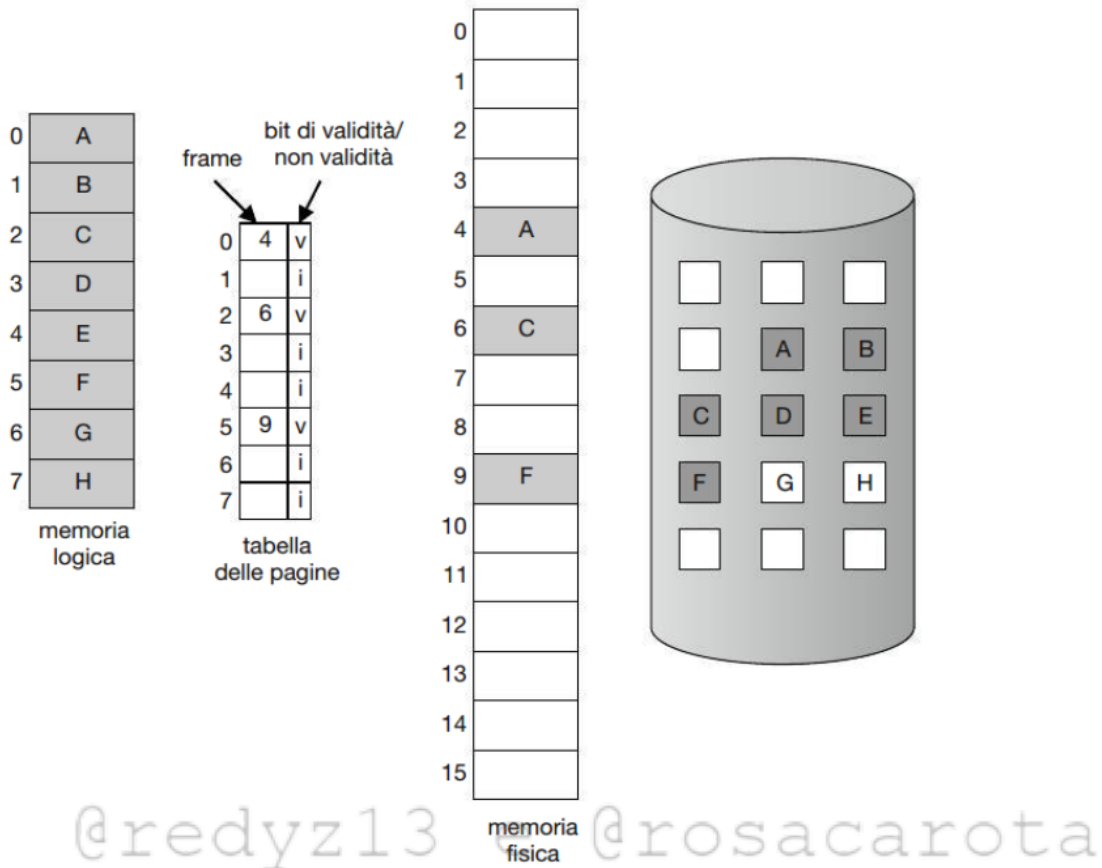
Inizializzato in partenza a 0

Esempio di tabelle delle pagine con bit di validità:



Se il processo tenta di accedere ad una pagina che non era stata caricata in memoria (bit di validità 0) → page fault

Tabella delle pagine quando alcune pagine non si trovano nella memoria centrale:



Meccanismi di ausilio alla pagina su richiesta

Tabelle delle pagine: questa tabella ha la capacità di contrassegnare un elemento come non valido attraverso un bit di validità oppure un valore speciale dei bit di protezione

Memoria ausiliaria: conserva le pagine che non sono presenti in memoria centrale. È costituita da un disco ad alta velocità detto dispositivo d'avvicendamento; la sezione del disco usata

per questo scopo si chiama **area d’avvicendamento o area di scambio (swap place)**

In generale il sistema operativo utilizza un file system ad alta velocità per la partizione di swap

Ad esempio il sistema operativo Linux utilizza una partizione distinta da quella del file system convenzionale in modo da velocizzare la gestione dei dati

La paginazione su richiesta può avere un effetto significativo sulle prestazioni generali di un sistema di calcolo

Gestione del page fault

Quando viene referenziata una pagina non in memoria, ci sarà un segnale di errore → page fault

La procedura di gestione del page fault è suddivisa in sei fasi:

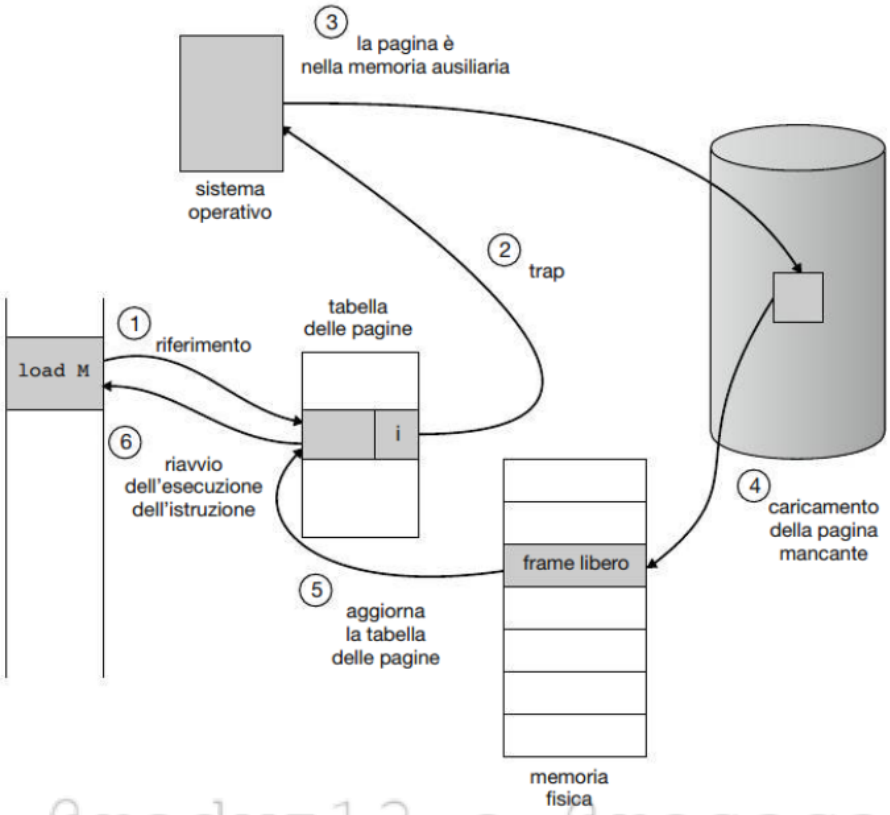
1. Si controlla la tabella interna del processo per stabilire se il riferimento alla pagina è valido o non valido
2. Se il riferimento non è valido si termina il processo, altrimenti se è valido ma la pagina non è presente in memoria, se ne effettua il caricamento in memoria
3. Viene individuato un frame libero, per esempio prelevandolo dall’elenco dei blocchi liberi
4. Si programma un’operazione sui dischi per trasferire la pagina desiderata nel blocco di memoria appena assegnato (swap in)
5. A swap in terminato, si modifica la tabella interna conservata con il processo e la tabella delle pagine per indicare che la pagina è disponibile nella memoria centrale
6. Si riavvia l’istruzione interrotta dal segnale di eccezione di indirizzo illegale. Il processo può accedere alla pagina

È addirittura possibile avviare l’esecuzione di un processo senza caricare inizialmente alcuna pagina in memoria

Quando il S.O. carica nel program counter l’indirizzo della prima istruzione del processo si genera subito un page fault. A questo punto il processo continua l’esecuzione e genera altri page fault finché tutte le pagine che gli servono non si trovano in memoria

Questo schema è noto come **paginazione su richiesta pura:** una pagina non viene mai trasferita in memoria se non viene richiesta

Fasi di gestione di un’eccezione di pagina mancante:



Se non è presente un frame libero:

- **Rimpiazzamento di pagina** - si cerca una pagina non utilizzata tra quelle in memoria e la si memorizza nella memoria ausiliaria (swap out)
  - Algoritmo
  - Prestazioni - l'algoritmo dovrà minimizzare il numero di page fault futuri

Quindi alcune pagine dovranno essere portate in memoria più volte

Prestazioni della paginazione su richiesta

La paginazione su richiesta può avere un effetto rilevante sulle prestazioni di un calcolatore

Calcolo del tempo di accesso effettivo:

- Tempo di accesso alla memoria → ma (normalmente varia da 10 a 200 nanosecondi finché non si verifica un page fault (cioè il tempo d'accesso effettivo è uguale al tempo di accesso alla memoria))

Probabilità di un assenza di pagina (page fault rate):

- $0 \leq p \leq 1.0$
  - Se $p = 0$ nessun page fault
  - Se $p = 1$, ogni riferimento genera page fault

Tempo di accesso effettivo (Effective Access Time - EAT)

$$\text{EAT} = ((1 - p) \times ma) + (p \times \text{tempo di gestione dell'assenza di pagina})$$

Per calcolare il tempo d'accesso effettivo occorre conoscere il tempo necessario alla gestione di un'assenza di pagina

Gestione di una assenza di pagina

1. Segnale di eccezione al sistema operativo
2. Salvataggio dei registri utente e dello stato del processo
3. Determinazione della natura di eccezione di pagina mancante del segnale di eccezione (se l'interruzione è dovuta o meno a un page fault)
4. Controllo della correttezza del riferimento alla pagina e determinazione della locazione della pagina nel disco
5. Lettura dal disco e trasferimento in un blocco di memoria libero:

a. Attesa nella coda del dispositivo fino a che la richiesta di lettura non è servita
b. Attesa del tempo di posizionamento e latenza del dispositivo
c. Inizio del trasferimento della pagina in un blocco di memoria libero
d. Durante l’attesa, assegnazione della CPU ad un altro utente
e. Segnale di interruzione emesso dal controllore del disco (I/O completato)
f. Salvataggio dei registri e dello stato dell’altro processo utente

6. Determinazione della provenienza dal disco dell’interruzione
7. Aggiornamento della tabella delle pagine e di altre tabelle per segnalare che la pagina richiesta è in memoria
8. Attesa che la CPU sia nuovamente assegnata a questo processo
9. Recupero dei registri utente, dello stato del processo e della nuova tabella delle pagine, quindi ripresa dell’istruzione interrotta

In sostanza il tempo di servizio dell’eccezione di page fault ha tre componenti principali:

- Servizio del segnale di eccezione di page fault
- Lettura della pagina dal disco
- Riavvio del processo

Prestazioni della paginazione su richiesta: un esempio

Considerando un tempo medio di servizio dell'eccezione di pagina mancante di 25 millisecondi e un tempo di accesso alla memoria di 100 nanosecondi, il tempo effettivo d'accesso in nanosecondi è:

$$EAT = (1 - p) \times 100 + p \times (25 \text{ millisecondi}) =$$
$$= (1 - p) \times 100 + p \times 25.000.000 =$$
$$= 100 + 24.999.990 \times p$$

Il tempo d'accesso effettivo è direttamente proporzionale alla frequenza delle assenze di pagina (**tasso di page fault**)

Ad es. se un accesso su 1000 accusa un'assenza di pagina, il tempo d'accesso effettivo è di 25 microsecondi

Se si desidera un rallentamento inferiore al $10\% : p < 0,0000004$

Cioè per mantenere a un livello ragionevole il rallentamento dovuto alla paginazione, si può permettere un'assenza di pagina ogni $2.500.000$ accessi alla memoria

Creazione ed esecuzione di processi

La memoria virtuale offre altri vantaggi per quel che riguarda la creazione ed esecuzione dei processi

Copiatura su scrittura (Copy-on-Write):

- Questa tecnica permette una creazione dei processi molto rapida e minimizza il numero di pagine che si devono assegnare al nuovo processo

Associazione dei file alla memoria (Memory-Mapped Files):

- Questa tecnica semplifica e velocizza gli accessi al file system trattando l'I/O su file tramite la paginazione in memoria centrale, invece che tramite le chiamate a sistema di **read()** e **write()**

Copiatura su scrittura (Copy-on-write)

La `fork` crea un processo figlio come duplicato del genitore

Nella versione originale `fork` creava per il figlio una copia dello spazio di indirizzi del genitore duplicando le pagine appartenenti al processo genitore

Considerando che spesso ad una `fork` segue una `exec`, questa copiatura risulta inutile

La tecnica di **copiatura su scrittura (Copy-on-Write - COW)** permette sia al processo genitore che al processo figlio di condividere inizialmente le stesse pagine di memoria

Se uno dei due processi modifica una pagina, solo allora la pagina è duplicata

COW permette una creazione di processi più efficiente e rapida

Quando una pagina deve essere duplicata, si attinge ad un pool di pagine libere utilizzate anche per gestire l'ingrandimento di strutture come pile o heap di un processo. Qui si attua l'**azzeramento su richiesta:** prima dell'allocazione le pagine vengono prima riempite di zeri e poi assegnate **(azzeramento su richiesta)**

Associazione dei file alla memoria

Consideriamo un accesso sequenziale in lettura o scrittura ad un file su disco, ogni operazione di lettura o di scrittura necessiterà di una chiamata al sistema e di un accesso al disco

L'associazione alla memoria permette di trattare l'I/O di file su disco come un generico accesso in memoria: una parte degli indirizzi virtuali è associata logicamente a un file. Questo si realizza facendo corrispondere un blocco del disco ad una pagina (o più pagine) di memoria

L'accesso a file avviene inizialmente tramite l'ordinario meccanismo di paginazione su richiesta, cui segue, però, il

caricamento in una pagina fisica di una porzione di file della dimensione di una pagina, leggendola dal file system

Le operazioni successive di lettura o scrittura si gestiscono come normali accessi alla memoria

Questa tecnica permette inoltre l’associazione dello stesso file alla memoria virtuale di più processi allo scopo di condividere dati. Le modifiche apportate da uno dei processi ai dati in memoria virtuale saranno visibili da tutti gli altri processi coinvolti.
Come funziona: la parte di memoria virtuale di ciascun processo punta alla stessa pagina di memoria fisica

Le chiamate a sistema per l’associazione alla memoria possono anche disporre della funzione di copiatura su scrittura, permettendo ai processi di condividere un file per la lettura, mantenendo però una propria copia dei dati da essi modificati

Associazione dei file alla memoria, esempio: