CAP. 8 : GESTIONE DELLA MEMORIA

Memoria dal punto di vista di OS

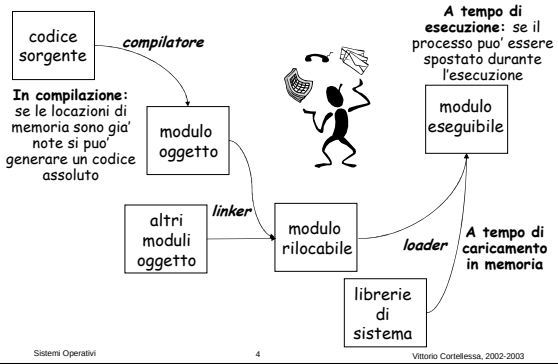
- Un ***programma*** per essere eseguito deve essere ***portato in memoria centrale*** ed utilizzare la memoria stessa per trattare con i dati di cui ha bisogno
- Il ***tempo di risposta*** di un sistema e l'***utilizzazione della CPU*** dipendono quindi ***anche*** dalla gestione della memoria centrale
- Le ***strategie*** adottabili per la gestione della memoria sono ***legate all'hardware*** di cui si dispone
- Ai fini di una ***gestione di memoria efficiente*** non importa come gli ***indirizzi*** di accesso a memoria siano stati generati, ma ***solo la loro sequenza***

Questione iniziale

Quando vengono ***scelti*** gli ***indirizzi di memoria*** nei quali verranno allocati istruzioni e dati di un processo???

Associazione (= binding)

istruzioni e dati → indirizzi di memoria



Ulteriori opzioni

- Caricamento dinamico
- Linking dinamico

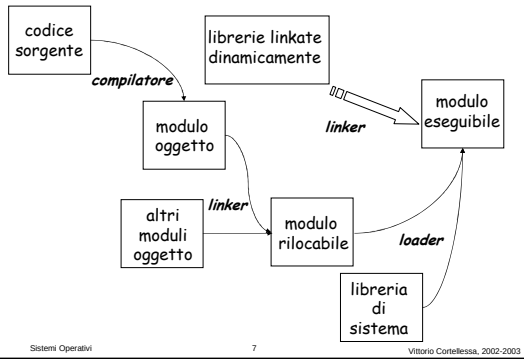
Caricamento dinamico

Una ***procedura*** non viene ***caricata*** finche' non viene ***invocata***

... <u>vantaggi</u>:

- Migliore ***utilizzo della memoria***
- Utile quando ***codici di grandi dimensioni*** sono necessari per manipolare ***casi molto infrequenti***

Linking dinamico …



… caratteristiche …

- Il linking viene ***posposto a tempo di esecuzione***
- Un piccolo pezzo di codice, ***stub***, viene piazzato ***al posto della routine*** ed utilizzato per localizzare e linkare la routine di libreria
- La routine puo' ***trovarsi gia' in memoria***, e quindi si deve solo verificare che sia nello spazio degli indirizzi del processo
- La routine puo' ***trovarsi sul disco***, e quindi un'operazione di ***caricamento dinamico*** deve essere effettuata

… e vantaggi:

- ***Non necessaria una copia*** della stessa routine in tutti i codici rilocabili che la invocano
- Collegamento automatico a ***nuove versioni*** della stessa libreria

Che significa virtualizzare la memoria?

Un processo occupa ***piu' spazio di quello disponibile*** per la sua allocazione

⇩

Tenere in memoria solo una parte delle ***istruzioni*** e dei ***dati*** di un processo : quelli ***necessari*** in ogni momento

E su questo ci torneremo ampiamente…

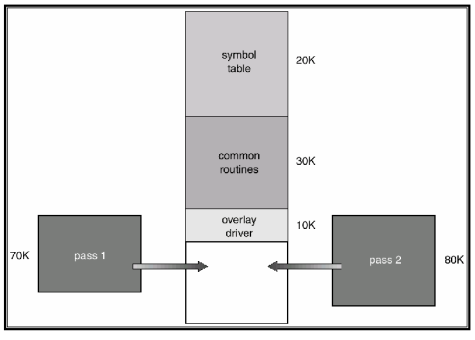
L'idea piu' semplice : *overlays*

Un programma viene suddiviso in ***moduli logicamente distinti***, overlays, che possono essere caricati in memoria in istanti diversi (***overhead di caricamento***)

La ***definizione*** e la ***gestione*** degli overlays e' ***eseguita dall'utente***, che si occupa quindi di implementare anche un gestore di overlays che ne sequenzi l'esecuzione

Vedremo tecniche piu' sofisticate…

Esempio di memoria insufficiente

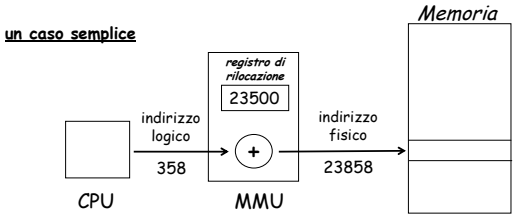


Il concetto di spazio di indirizzi

- Un processo si riferisce ad uno spazio di indirizzi logici, che e' associato ad uno spazio di indirizzi fisici
  - ***Indirizzo logico (o virtuale)*** : generato dalla CPU
  - ***Indirizzo fisico*** : quello visto e trattato dalla unita' di memoria vera e propria
- Nel ***binding a tempo di esecuzione*** indirizzi logici e fisici differiscono… chi si occupa di operare la traduzione da logici a fisici?

Memory-Management Unit (MMU)

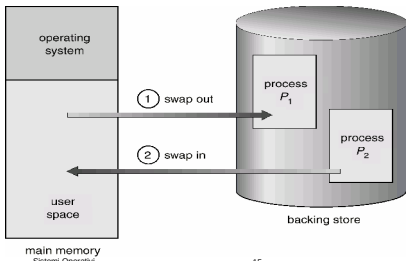
- I ***programmi utente*** trattano con ***indirizzi logici*** e non vedono mai i veri indirizzi fisici
- ***MMU*** e' il congegno hardware che mappa indirizzi logici a indirizzi fisici



Il meccanismo interno alla MMU puo' essere ***molto piu' complesso*** di un semplice registro di rilocazione!

Swapping…

Un processo puo' essere ***portato via dalla memoria*** (swapped out) ad una memoria secondaria, ed in seguito ***riportato in memoria*** (swapped in) per proseguire la sua esecuzione



Ricordate il medium-term scheduler?

… e alcune considerazioni

- Il ***tempo di swapping*** e' proporzionale all'ammontare di memoria swapped
- Ha senso solo se si possono ***tenere in memoria piu' processi alla volta*** e quindi lo swapping non avviene ad ogni context switching e non coinvolge tutta la memoria utente
- E' come se la ***ready queue*** proseguisse pure ***su disco*** (***coda di input***)

… e alcune considerazioni

- Il criterio di swapping si puo' basare, per esempio, sulla ***priorita' dei processi***
- A causa del suo costo elevato, si puo' pensare ad uno ***swapping non attivo sempre***
- Problemi relativi a ***processi swapped out*** mentre stavano ***su code di dispositivi di I/O*** (scrittura in buffer di I/O non piu' validi)

MA SOPRATTUTTO …

Con ***binding a tempo di esecuzione*** un processo puo' essere anche ricaricato in una zona diversa della memoria centrale

Metodologie fondamentali di allocazione della memoria

- Allocazione contigua
- Paginazione
- Segmentazione

Allocazione contigua

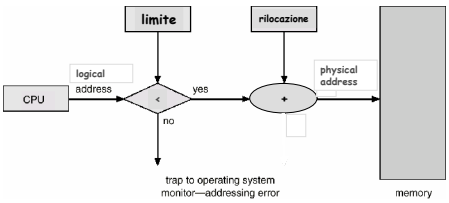
La memoria e' solitamente suddivisa in due "zone":

- ***Sistema operativo*** (se necessario con spazio per l'interrupt vector)
- ***Processi utente***

Anche ***OS*** stesso puo' avere ***dimensione variabile*** in memoria : ex. device drivers

Partizione singola

Registri di ***rilocazione*** (base) e ***limite*** usati per proteggere OS dai processi utente



Rilocazione : il piu' piccolo indirizzo fisico
Limite : il piu' grande indirizzo logico

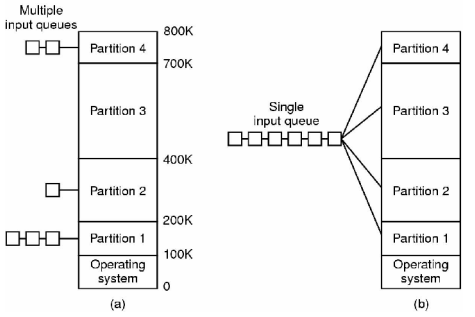
Partizioni multiple : piu' processi in memoria allo stesso tempo

IDEA BANALE:
Suddivisione della memoria in un numero fisso di ***partizioni*** tutte ***aventi la stessa dimensione***

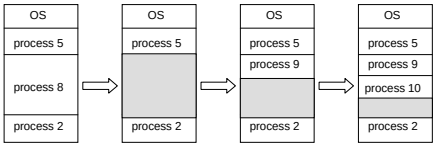
⇩

- ***Grado di multiprogrammazione*** prefissato
- ***Taglia massima*** di processo che puo' essere eseguito

… ma si puo' anche pensare a partizione fisse di dimensioni diverse

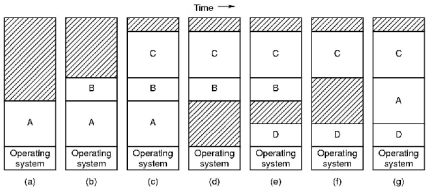


Un'idea meno banale: partizioni multiple a dimensione variabile



- ***Buco*** – blocco di memoria disponibile
- Quando un processo arriva, gli viene ***allocato un buco di memoria grande abbastanza*** per sistemare il processo

Un'idea meno banale: partizioni multiple a dimensione variabile



- ***Buchi di dimensione varia*** vengono originati lungo la memoria
- OS deve mantenere ***traccia*** delle ***partizioni allocate*** e dei ***buchi***

Tabella dei buchi : collocazione e taglia di ogni buco
Coda di input : processi pronti che non sono in memoria

… e qualche considerazione …

• Se il prossimo processo candidato ad entrare in memoria non trova un buco di dimensione adeguata alla sua taglia, la ***strategia di ingresso in memoria*** puo' anche considerare il processo seguente, e cosi' via…

• La ***tabella dei buchi*** viene ***aggiornata*** ogni volta che un processo entra o esce dalla memoria

• Quando un processo esce dalla memoria puo' essere un momento opportuno per ***dare uno sguardo alla coda di input***

Un problema generale

Allocazione dinamica della memoria : come soddisfare una richiesta di ***taglia n*** da una tabella/lista di buchi?!

- **First-fit**: Alloca il ***primo*** buco che e' grande abbastanza
- **Best-fit**: Alloca il ***piu' piccolo*** buco che e' grande abbastanza
- **Worst-fit**: Alloca il ***piu' grande*** buco che e' grande abbastanza

Parametri per la valutazione di una strategia :

- ***velocita'*** di allocazione (*first- fit*)
- ***utilizzazione*** della memoria

Un problema di utilizzazione della memoria: la frammentazione

Quando la differenza tra memoria richiesta e buco disponibile e' molto piccola vengono allocati blocchi leggermente piu' grandi perche' sarebbe oneroso (e inutile?) tenere traccia di tutti

⇩

Frammentazione interna – distribuzione di spazio libero ***all'interno di blocchi occupati***

Esiste spazio totale in memoria per soddisfare una richiesta, ma tale spazio non e' contiguo

⇩

Frammentazione esterna – distribuzione di spazio libero ***tra blocchi occupati***

⇨

La ***frammentazione esterna*** e' piu' comune e ***piu' grave*** in termini di ***utilizzazione di memoria***

In media viene ***sprecato un terzo dello spazio***

First-fit e ***best-fit*** sono migliori di ***worst-fit*** in quanto a velocita' di allocazione e utilizzazione di memoria

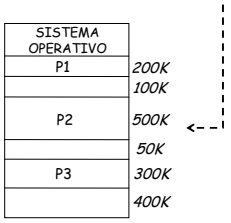
Una soluzione immediata : ***la compattazione***

Spostare i contenuti della memoria per piazzare tutta la memoria disponibile in ***un unico buco grande***

(Naturalmente e' possibile solo se ***l'associazione degli indirizzi e' a tempo di esecuzione***)

Compattazione

Varie strategie di compattazione (esempi)



Quanto spesso compattare? Se lo swapping e' gia' parte di OS, il codice per la compattazione e' minimo