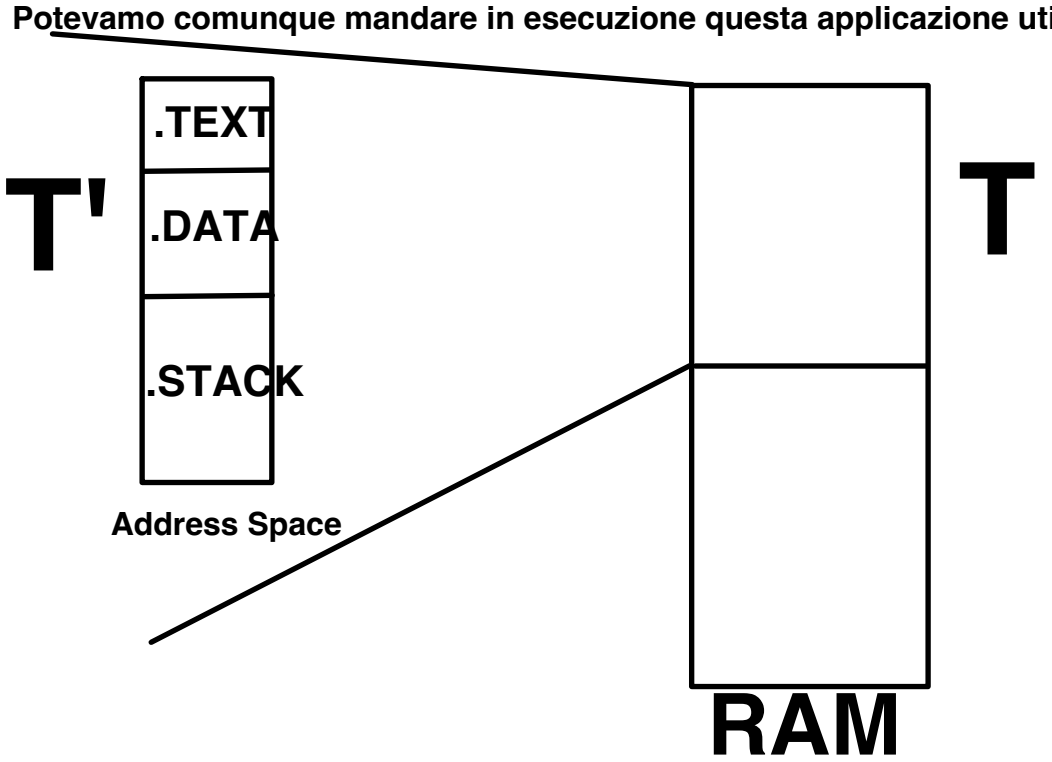
Overlay

giovedì 11 maggio 2023 12:06

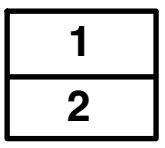
Overlay

L'overlay era una tecnica che veniva utilizzata da programmatori esperti che lavoravano per risolvere questo problema:
Abbiamo un'applicazione il cui "address space" con taglia T' non entra all'interno della massima partizione della RAM con taglia T.
Ma T' > T.
Potevamo comunque mandare in esecuzione questa applicazione utilizzando la partizione di taglia T?



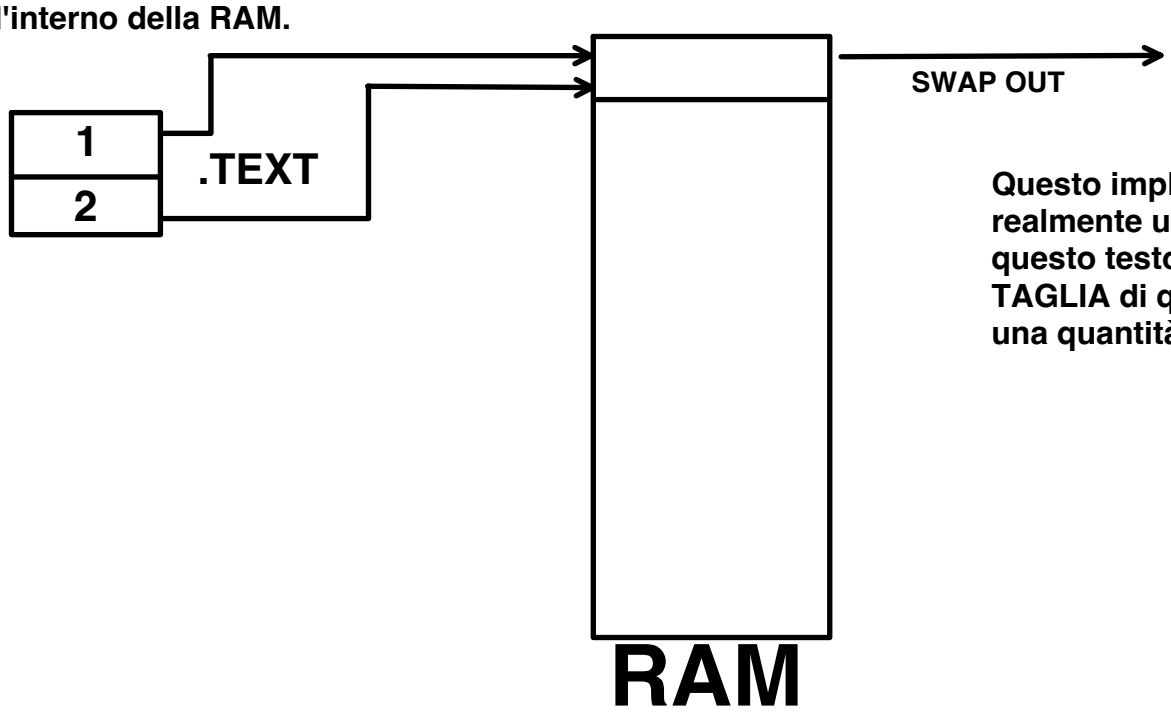
È vero che il mio address space ha una taglia T' ma è altrettanto vero che questo address space è strutturato in una zona .TEXT, in una zona .DATA e magari sotto una zona .STACK.
<u>È sempre vero che tutte queste informazioni all'interno dell'address space mi serve che siano tutte realmente caricate all'interno della memoria per portare in esercizio questa applicazione</u>?
Posso pensare di avere questo:

Dato il testo, magari inizialmente per eseguire mi serve solo una parte della zona 1 del testo e dopo aver fatto alcune attività mi serve la zona 2. Quindi una parte delle istruzioni e delle funzioni che fanno parte di questa applicazione.



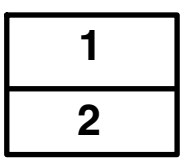
.TEXT

Bene, io potrei avere all'interno della RAM caricare soltanto una di quelle parti. Il che implica dire che sto caricando all'interno della RAM una FRAZIONE delle informazioni che caratterizzano il mio address space.
Quando ho finito di eseguire le attività che sono correlate a quella frazione vado a sostituire le informazioni che ho all'interno della ram, caricando la zona 2.
Sto caricando inizialmente una frazione di questo testo e successivamente un'altra frazione di questo testo all'interno della RAM.



Questo implica dire che non sto utilizzando realmente una quantità di RAM per ospitare questo testo che è pari esattamente alla TAGLIA di questo testo, anzi, sto utilizzando una quantità di RAM inferiore.

Questo procedimento vale anche per la sezione dati dell'address space.



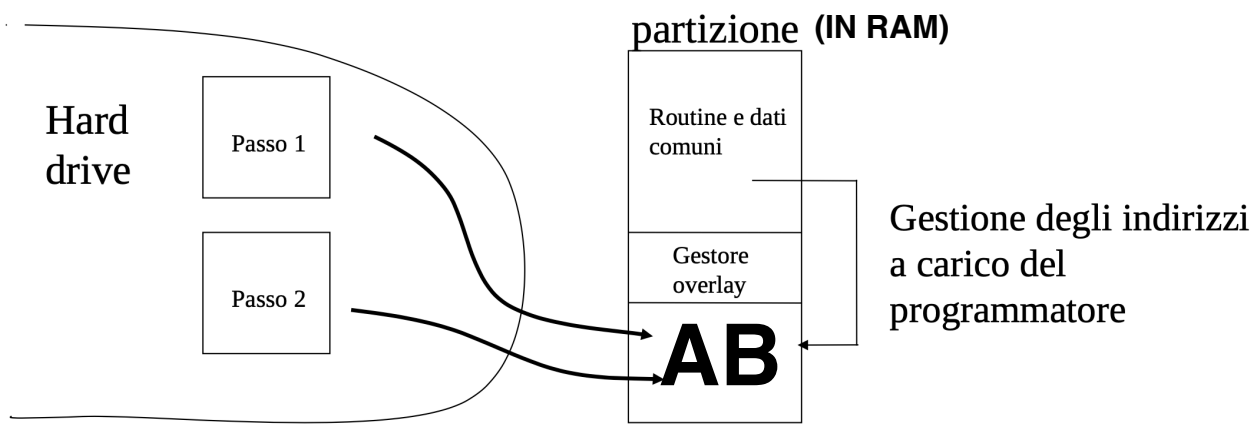
.DATA

Anche qui sto utilizzando una quantità di RAM che non è necessariamente pari alla taglia dei dati che sto gestendo durante l'esecuzione dell'applicazione.

Questo è quello che succede esattamente nell'OVERLAY.

Al programmatore è lasciato il compito di gestire il caricamento dinamico all'interno della zona di memoria che eventualmente è riservata per l'address space. In un punto dell'address space è possibile caricare il passo 1 di un'applicazione, quindi testo e dati che servono inizialmente, e poi caricare all'interno della stessa zona dell'address space (e quindi ovviamente anche della RAM dove questo AB è ospitato) un passo2.
Per far questo dovrò mantenere in RAM dall'inizio alla fine dell'applicazione le routine e i dati comuni in tutte le fasi d'esecuzione, e poi dovrò avere all'interno del mio address space il gestore dell'overlay, quindi colui che ad un certo punto viene chiamato dalle routine che dice di caricare nella zona dell'address space le informazioni richieste (passo 1 e passo 2).

- processi con taglia superiore a quella della massima partizione non possono essere attivati dal sistema operativo
- nella tecnica **overlay** al programmatore è lasciato il compito di gestire tale situazione



Queste informazioni da caricare le posso prendere anche dal file system e caricarle dinamicamente all'interno dell'address space, in cui una zona dell'address space, la zona AB, è messa lì per ospitare informazioni, COME UN BUFFER.
E lì ci andranno informazione che ad un certo istante di tempo sono il passo 1 e ad un altro istante di tempo il passo 2.
Così la mia applicazione può andare in esercizio anche non avendo RAM sufficientemente ampia per ospitare tutte queste informazioni, ma anche scenari in cui in partizionamento statico, mi genera una partizione della RAM, dove non posso mantenere tutti e due i passi.
Queste informazioni possono essere sia routine che dati.

routine e dati comuni + gestore overlay + passo 1 = **overlay A**
routine e dati comuni + gestore overlay + passo 2 = **overlay B**

Si tratta di una tecnica che permette ad un computer di eseguire dei programmi che richiedono una quantità di memoria centrale superiore a quella effettivamente disponibile.
Questa tecnica di gestione della memoria centrale era usata nei sistemi d'elaborazione che non facevano uso della multiprogrammazione ed è ancora usata nei sistemi d'elaborazione integrati in altri dispositivi, poichè in questi sistemi la quantità di memoria centrale a disposizione è molto limitata.
Per fare uso di questa tecnica, un programma dev'essere suddiviso manualmente in segmenti di codice denominati overlay ed organizzati gerarchicamente, con un segmento principale che durante l'esecuzione risiede sempre in memoria, ed un certo numero di segmenti di overlay ciascuno dei quali, durante l'esecuzione del programma, viene letto dalla memoria secondaria e caricato in memoria centrale solo quando è richiesto, dopo di che, l'area di memoria da esso utilizzata (overlay area) può essere assegnata ad altri segmenti di overlay del programma.

Questo è correlabile al concetto di DLL. Quello che mi serve dinamicamente lo carico all'interno dell'address space e lo utilizzo.
Ma tutto questo prescinde dal concetto di binding dinamico, perché io potrei avere il caricamento di informazioni, tali per cui io potrei sapere qual è la posizione in RAM (in maniera statica) di tutta la PARTIZIONE.