MEMORY MANAGEMENT

Diapositiva 1
la gestione della memoria serve a suddividere la memoria principale in parti necessarie al sistema operativo e parti necessarie ai programmi per la loro esecuzione.

Diapositiva 2
la gestione deve essere fatta in modo da soddisfare i seguenti principi:
-Rilocazione, perchè i processi entrano e escono dalla memoria.
-Protezione, in quanto i dati di un processo non devono subire interferenze indesiderate da parte di altri processi.
-Condivisione, alcune informazioni potrebbero dover essere condivise tra più processi.
-Organizzazione Logica, a livello di indirizzamenti, per esempio con tecniche come la segmentazione.
-Organizzazione Fisica, tra memoria principale e memoria secondaria oppure tra i diversi blocchi di memoria.

Diapositiva 3
ci sono diverse tecniche più o meno sofisticate per tenere traccia di quali zone sono allocate e non.
se un processo chiede di allocare della memoria (es con una malloc) non è il SO che se ne occupa, bosognerebbe fare una sytem call che è uno spreco di tempo magari per allocare 20 byte! quando un processo viene attivato gli viene allocata una zona di memoria relativamente grande cosicchè se il processo deve allocare della memoria lo fa direttamente lui in una zona di questa memoria che gli è stata assegnata. questa zona è lo HEAP, gestito da un gestore di memoria interno al processo. nella memoria dedicata al processo ci sono diverse zone, per esempio la zona con il programma effettivo, le istruzioni che lo compongono, uno stack per gestire gli interrupt e altre zone tra cui proprio quello heap. ovviamente il SO non sa a priori di quanta memoria ha bisogno il processo, quindi in fase di elaborazione se il processo necessita di uno heap più esteso lo chiede al SO. viene assegnata a pezzi "grossi" della memoria in più, in genere multipli di 4KB, non una cella per volta...

Diapositiva 4
se tengo pochi processi in memoria magari a un certo punto saranno tutti in attesa di I/O e il processore sarà inutlizzato

Diapositiva 6
Le prime semplici tecniche per la gestione della memoria sono state quelle di partizionamento della memoria stessa o di paginazione e segmentazione. si è poi sviluppato il concetto di memoria virtuale, ma vediamo prima, anche se obsolete, quelle vecchie tecniche che non coinvolgevano la memoria virtuale.

Diapositiva 7
cioè devono essere fatti in modo che solo una porzione di essi risiede nella memoria principale al momento in cui iniziano ad essere eseguiti e se poi c'è bisogno di un modulo non presente in memoria viene caricato sovrascrivendo dati già presenti.

Diapositiva 8
frammentazione interna: i programmi non usano effettivamente tutta la memoria che gli è stata alloccata quindi in mezzo alla memoria si formano delle zone inutilizzate che sono quindi uno spreco.
anche se con le dimensioni diverse di partizionamento si aumenta un pochino la flessibilità di questa tecnica, il partizionamento fissa rimane cmq una tecnica svantaggiosa per due motivi principalmente:
1-fissare un partizionamento definisce a priori un numero di partizione che limita il numero di processi attivi nel sistema ad un certo momento
2-i piccoli job, che in un sistema operativo sono numerosi, non sfrutteranno mai in maniera efficiente la memoria e ci sarà cmq nella maggioranza dei casi spreco di memoria e relativa frammentazione interna.

Diapositiva 9
utilizzare partizioni fisse ma di diverse dimensioni diminuisce un po' il fenomeno della frammentazione interna in quanto si può scegliere il blocco di dimensioni più adatte a limitare lo spreco di memoria. anche questo cmq è molto inefficiente.

Diapositiva 10
quando tutte le partizioni fossero occupate da processi non ready lo schedulatore dovrà decidere quale processo swappare su disco per liberare spazio per un nuovo processo ready.

MEMORY MANAGEMENT

Diapositiva 11
dal punto di vista di una singola partizione questa tecnica minimizza la frammentazione all'interno della stessa. dal punto di vista dell'intero sistema però non è così efficiente avere una coda per ogni partizione. si preferisce quindi averne una unica che selezioni sempre con il criterio di scegliere la partizione più piccola che contiene il processo. nel momento in cui ci sia necessità di scaricare un processo su disco per liberare la memoria si sceglie in genere di scaricare la partizione più piccola che contiene il processo in arrivo, anche se si possono considerare altri fattori come la priorità, l'I/O che il processo farà, ecc.

Diapositiva 12
A volte la compattazione può essere irrealizzabile, come ad es. in C o C++ in cui la malloc non si occupa della soluzione a questo problema. effettuare uno spostamento significa anche controllare ovunque ci siano dei puntantori capire a cosa puntano e in caso modificare anche quelli!!! è troppo complicato.

Diapositiva 14
se ora un processo più grande richiede memoria non può essere eseguito!
Questa tecnica sembra partire bene con il suo principio di assegnare esattamente la memoria necessaria ma prima o poi anche questa prota ad avere tanta frammentazione.

Diapositiva 15
per limitare il più possibile la frammentazione esterna il SO deve può usare diverse tecniche per allocare la memoria ai processi.
deve leggere tutta la memoria per trovare la porzione best-fit. inoltre si creano tantissime piccole porzioni in giro che saranno sicuramente troppo piccole per accogliere altri processi. La frammentazione quindi aumenta più che negli altri algoritmi e alla lunga la compattazione deve essere fatta più spesso.

Diapositiva 16
siccome first-fit tende a riempire la parte iniziale della memoria di piccole partizioni libere, queste ogni volta vengono esaminate rallentando un pochino il procedimento. nonostante ciò questo è l'algoritmo più veloce e più efficiente.

Diapositiva 17
compaction più frequente.

Diapositiva 19
il partizionamento fisso limita il numero di processi attivi mentre quello dinamico ha lo svantaggio oneroso della compattazione. il buddy system è un compromesso tra i due.
gli allocatori funzionano alla perfezioni con blocchi di multipli di 4KB perchè questa dimensione è anche supportata dall'hardware.

Diapositiva 20
in generale i blocchi di memoria hanno dimensione 2^k, con $L \leq K \leq U$, dove 2^L è il blocco allocato di dimensione più piccola e 2^U il blocco allocato di dimensione più grande, che infatti in genere corrisponde all'intera memoria disponibile.

Diapositiva 22
buddy: "amichetti". due blocchi possono essere aggregati solo se sono buddies, o meglio se sono fratelli (possiamo vedere lo splitting come un albero binario)
anche se sono stati richiesti 100KB se ne assegnano 128 perchè si alloca sempre per potenze di 2.
due porzioni derivanti dallo stesso dimezzamento sono due "buddies"

Diapositiva 24
due nodi fratelli sono buddies tra loro. se sono entrambi foglia allora almeno uno dei due deve essere allocato altrimenti sarebbero riuniti in un blocco non allocato maggiore.
Anche se le tecniche attuali hanno soppiantato il buddy system questo si è rivelato un ottimo compromesso tra il partizionamento fisso e quello dinamico

MEMORY MANAGEMENT

Diapositiva 26
i requisiti per i processi come già detto sono:
-la rilocazione (dei processi e dei dati)
-la protezione
-la condivisione
-l'organizzazione logica
-l'organizzazione fisica

anche per via della compattazione le posizioni de programmi all'interno della memoria variano.

Diapositiva 28
es. le librerie dinamiche condivise, .dll (dynamic link library) in windows .so (shared object) in unix

Diapositiva 31
quindi l'indirizzamento dei programmi deve poter gestire in maniera semplice e veloce queste variazioni.

Diapositiva 32
Si distinguono 3 tipi di indirizzamento:
-FISICO
-LOGICO
-RELATIVO
L'indirizzamento fisico sempre meno utilizzato, si può trovare a volte in alcune parti del kernel

Diapositiva 33
Base Register: contiene l'indirizzo iniziale del processo che è in memoria principale e che è attualmente nello stato running.
Bounds Register: contiene l'indirizzo della locazione finale del programma.

Diapositiva 35
i valori del base register e del bound register limitano la porzione indirizzabile implementando così anche il meccanismo di protezione per processi esterni.

Diapositiva 36
suddivisione in PAGINE del PROCESSO e in FRAMES della MEMORIA.
in questo modo in ogni frame non c'è mai spazio sprecato. l'unica eccezione è il frame a cui è stato associata l'ultima pagina di processo che può avere uno spreco, ma comunque la frammentazione interna è estremamente ridotta, se si considera che i frame presi da un processo sono moltissimi e quindi in proporzione il numero di frame non interamente occupati sono molti di meno. la frammentazione esterna invece con questo meccanismo è del tutto eliminata perchè la memoria è divisa in sezioni fisse che la impediscono! (come nel partizionamento fisso).
il SO si occupa di assegnare un certo frame della memoria ad una certa pagina del processo. nei sistemi a 32 bit moderni frame e pagine sono 4KB

Diapositiva 37
Le pagine 0,1,2,3 del processo A sono state allocate nei frame 0,1,2,3 della memoria.

Diapositiva 38
si possono assegnare ad un processo anche pagine che sono distanti tra loro sfruttando il meccanismo dell'indirizzamento logico e utilizzando semplicemente un numero maggiore di base register, quindi come già detto si elimina il problema della frammentazione esterna. in questo caso il base register consiste nel numero di frame page.

Diapositiva 39
il SO tiene sempre traccia di quali sono le pagine libere in memoria.
Questo meccanismo in sostanza è molto simile al partizionamento fisso, reso però più efficiente dal fatto che le partizioni sono molto piccole e che un processo può occupare anche partizioni non contigue e quindi si evita la frammentazione sia interna che esterna

Diapositiva 40

dal 386 in poi la segmentazione semplice è via via sparita. non esistono più ad oggi sistemi che supportano la sola segmentazione ma esistono ancora sistemi che supportano solo la paginazione, come linux. Windows invece ad esempio usa una tecnica mista detta "segmentazione paginata".
anche in questo caso il SO tiene traccia di una serie di tabelle dei segmenti per i diversi processi e di una tabella per i segmenti liberi
così come la paginazione era una evoluzione migliorata del partizionamento fisso, la segmentazione è simile al partizionamento dinamico in cui però le partizioni usate possono non essere contigue. la segmentazione quindi, proprio come il partizionamento dinamico, elimina la frammentazione interna ma causa ancora quella esterna anche se in questo caso la riduce notevolmente.

Diapositiva 41
CREAZIONE DELL'INDIRIZZO LOGICO A PARTIRE DALL'INDIRIZZO RELATIVO:
E' conveniente che le dimensioni delle pagine e dei frames siano potenze di 2. In questo caso le pagine sono di 1 KB (cioè 1024 byte=2^{10}) e gli indirizzi sono di 1 bit. Per puntare tutte le locazioni della pagina quindi abbiamo bisogno di 10 bit e rimangono 6 bit per puntare a 2^6=64 pagine. L'indirizzo è 1052, quindi la locazione sarà relativa alla seconda pagina (la numero 1 poichè la numerazione inizia da 0). l'offset all'interno di questa seconda pagina sarà pari a 1502-1024=478. Se ora si scrive in binario il numero di pagina seguito dall'offset si ottiene un indirizzo identico a quello di partenza.
Questo è il grande vantaggio dell'usare dimensioni pari a potenze di 2.

Diapositiva 42
TRADUZIONE DA INDIRIZZO LOGICO A INDIRIZZO FISICO:
-si legge nei primi N bit il numero di pagina.
-si legge nella page table l'indirizzo di partenza relativo a quella pagina
-si costruisce l'indirizzo fisico facendo seguire ai bit letti nella page table l'offset

Diapositiva 43
TRADUZIONE DA INDIRIZZO LOGICO A INDIRIZZO FISICO:
-si estrae dai primi N bit il numero di segmento
-si legge nella tabella l'indirizzo iniziale per quel segmento (la tabella in questo caso contiene due voci: una che indica la lunghezza del segmento e una che indica l'indirizzo iniziale)
-si SOMMANO l'indirizzo base e l'offset.