PAGINAZIONE

- Tecnica gestione della meoria usata per la multiprogrammazione + porzioni di memoria non contigue + processi non interamente in memoria.

- Indirizzi logici: numero pagina + offset. Numero pagina passa per la tabella delle pagine per avere la traduzione. Una tabella per processo. Se e piccola allora puo essere tenuta interamente in MMU, else si tiene solo l'indirizzo di dove si trova la tabella nella memoria centrale. In questo modo per fare un accesso in memoria centrale si devono fare 2 acessi, uno alla tabella delle traduzioni e uno all'indirizzo fisico traddoto da quello logico. Si introduce in questo caso la TLB, mem associativa, veloce, piccola, costosa che tiene le traduzioni recenti assieme all'ASID per poter identificare il spazio d'indirizzamento del processo a cui appartiene questa traduzione, in modo da poter tenere piu traduzioni per processi diversi.

- Non c'e frammentazione esterna ma c'e una minima frammentazione interna poiche non tutti i programmi sono grandi quanto un multiplo della dimensione di una pagina / frame, quindi parte dell'ultimo frame usato per il processo puo non essere completamente utilizzato. Questa situazione si puo risolvere facendo diventare la dimensione delle pagine piu piccole ma in questo modo scade anche la performance della paginazione dovendo caricare piu pagine che prima quando le pagine erano piu grandi, quindi non ci sta.

- Roba di protezione quindi bit R, W, X per capire cosa si puo fare con ogni pagina. Bit di validida / invalidita per capire se la pagina in questione e o no caricata in memoria centrale dalla zona di swap. Se si cerca di fare un'accesso a una pagina col bit di validita a 0 allora si avra una trap hardware (page fault) che dovra essere gestita poi dal sistema operativo caricando la pagina in questione nella memoria centrale. Se non c'e posto ci sono diversi modi per scegliere una pagina da sostituire con questa che dobbiamo caricare. Ci sono diverse politiche per scegliere cosa scaricare nella zona di swap, tipo LRU, LFU, ecc.

- Pagine condivise in caso di codice comune tipo librerie. Si possono condividere frame tra processi mappando una pagina allo stesso frame in tutte e due i processi. Il codice non dev'essere automodificante e deve trovarsi nella stessa locazione logica in tutti i processi, altrimenti non si puo autoreferenziare.

- Tipi di paginazione
    - **Gerarchica**. Si puo paginare la tabella della pagine visto che si trova in memoria comunque. In questo modo possiamo avere tipo un'indirizzo: p1 che dice la tabella delle pagine da usare, poi p2 per offset nella tabella scelta poi d offset nella pagina.
    - **Tabella delle pagine con hashing**. Si applica una funzione di hashing all'indirizzo logico e poi si cerca il risultato in una tabella di hashing. Questo risultato puo essere associato a una lista di overflow che dev'essere poi scandita per trovare la pagina richiesta
    - **Tabella delle pagine invertita**. Una sola tabella globale che per ogni frame tiene l'ASID del processo a cui e legata insieme al numero logico della pagina in quel spazio d'indirizzamento e poi i ibit di protezione.

SEGMENTAZIONE

- Serve per tipizzare le varie porzioni di memoria. Nella paginazione non funziona. Anche questa una tecnica di gestione della memoria centrale che consente la multiprogrammazione e di utilizzare porzioni di memoria non contigue per un processo + non tenere l'intero processo in memoria, ma soll i segmenti che servono in quel momento o immediato futuro.

- Rispetto alle pagine i segmenti hanno dimensione variabile e quindi c'e il problema della frammentazione esterna visto che possono rimanere vuoti dei spazi in cui non posso inserire un'altro segmento. La frammentazione interna non c'e visto che i segmenti hanno dimensione fatta per tenere essattamente quello che serve, non di piu. La frammentazione esterna si puo risolvere con la compattazione della memoria.
    - Ci sono piu modi per inserire un nuovo segmento: first fit (il primo buco in cui sta), best fit (il piu piccolo in cui ci sta), worst fit (il piu grande). Per colpa del fatto che è molto improbabile che si trovi un buco di dimensione esatta, apparre la frammentazione esterna.

- Tabella dei segmenti, una per processo, indirizzo logico come nella paginazione. Traduzione fatta dalla MMU, che a ogni traduzione controlla anche dei bit di protezione per impedire accessi a segmenti inesistenti, accessi illegali, offset oltre il limte del segmento. Genera in questi casi una trap hardware (segfault).

- La MMU tiene o la tabella dei segmenti o l'indirizzo come nella paginazione. TLB stessa spiegazione con poi l'ASID.

- Essendo che tutti i segmenti sono tipizzati e quindi sono usati allo stesso modo, si associa a ogni segmento dei bit di protezione (R, W, X).

- Condivisione semplice come nella paginazione.

- Segmentazione con paginazione
    - per evitare la frammentazione esterna, identificare facilmente i frame liberi e che frame caricare si usa la paginazione. Dalla segmentazione si usano le cose riguardanti la protezione, il controllo fatto ad ogni accesso, e la tipizzazione + condivisione di un segmento tipizzato tipo codice idk.
    - la memoria logica viene divisa in segmenti che vengon divisi a loro volta in pagine inserite poi nei frame. Quindi un'indirizzo logico e tipo numero di segmento, pagina nel segmento, offset nella pagina. quelli fisici vabbe frame + offset.

MEMORIA VIRTUALE

Memoria virtuale permette di non tenere interamente un processo in memoria per esegurilo, condividere memoria, usare memoria non contigua.

Paginatore svolge il compito di gestire le page fault caricando nel caso di una pagina valida, la pagina in un frame libero. Il tempo d'accesso alla memoria dipende dalla frequenza di page fault. Essendo che dura tanto gestire un page fault, vogliamo che siano rari.

La chiamata **fork** duplica lo spazio d'indirizzamento del processo da cui viene chiamata. Con la memoria virtuale pero possiamo fare delle economie abbastanza grandi facendo copy on write, che quindi si carica una pagina nuova su cui si mettono le cose modificate sono nel caso di una vera modifica. Le cose non modificate non necessitano pagine nuove, sarebbe uno spreco.

Quando **non ci sono frame liberi** in cui caricare una pagina si puo sostituire una pagina gia esistente con quella nuova. La pagina ideale da sostituire sarebbe quella che non sara usata per tanto tempo, ma non si puo sempre sapere a priori. Quando ci sono processi che stanno aspettando in coda di IO, e possibile dare le sue pagine ai processi che neccessitano. Per non fare ulteriormente il processo di scrittura nell'area di swap, si puo scrivere roba nell'area di swap mentre il paginatore e a riposo, per non farlo dopo.

Soluzione per trovare la vittima FIFO: si selezione la pagina piu vecchia. Il problema con questa soluzione e che anche se la pagina e da tanto nella memoria, non vuol dire che essa non sia usata spesso. Questa soluzione soffre dell'anomalia di Belady: aumentando il numero di frame del processo, la frequenza di page fault cresce.

Soluzione LRU. Si fa o con timestamp piu basso o con un stack dove si mette in cima il processo usato di recente e si elimina quello piu in basso. Essendo che ste soluzioni sono pesanti si usano anche LRU approssimati tipo bit di riferimento, piu bit di riferimento e periodicamente si shifta a destra e si mette il bit di riferimento come MSB (il numero piu piccolo verra poi eliminato), algoritmo della seconda possibilita (se si trova una pagina con 0 si tira fuori, se ha 1 si mette a 0 dando una seconda possibilita),

Algoritmi bassati su conteggio: la pagina con uso minore è la vittima.

Algoritmi di allocazione - c'e quella omogenea che per m frame e n processi da m / n a ogni uno. Poi l'allocazione proporzionale che da numero di frame proporzionale col carico di lavoro ecc.

Thrashing - quando un processo sta piu tempo a paginare che a eseguire siamo in una situazione di thrashing. Una delle soluzione e usare il modello working set che definisce la localita di un processo come le pagine usate nei ultimi n accessi. Si puo usare sta cosa anche per la prepaginazione in modo che quandi si carica o un processo sospeso si riprende si carica il suo ws. Un'altro modo e quello basato sulla frequenza del page fault. Se questo sale sopra una certa soglia allora gli si allocano di piu frame, se scende sotto una certa soglia allora gli si deallocano frame. Se non ci sono frame si sospende.

PROTEZIONE

Protezione - sicurezza delle risorse fisiche e logiche contro accessi non autorizzati di utenti, processi. E fornita da un meccanismo che controlla l'accesso alle risorse, che deve fornire i mezzi per stabilire le regole e applicarle. Riceve le richiesta dei processi e decide se autorizzarle.

Ogni risorsa ha un id e un'insieme di operazioni. I processi dovrebbero poter accedere solo alle **risorse minime necessarie** alla loro computazione (non dovrebbero poter accedere a tutte le risorse esistenti). Ogni processo opera in un suo **dominio di protezione**, che definisce le risorse a cui puo accedere e le operazioni legali.

L'**associazione tra processo e dominio** puo essere statica oppure dinamica, quindi rimanere in un dominio o cambiare dominio.

La **matrice d'accesso** fornisce un meccanismo per specificare le politiche. Ha come righe i **domini** e come colonne le **risorse**. La cella i,j mostra le operazioni lecite che un processo di dominio i puo fare su una risorsa j. Anche i domini possono essere visti come risorse. Nel caso j e un dominio e in i,j ho **switch**, allora vuol dire che posso cambiare il dominio di un processo di dominio i in un processo appartenente al dominio di protezione j.

Operazione di copia - copia l'autorizzazione che un dominio ha su un oggetto a un'altro dominio, con o senza diritto di propagazione. (Copia)

Operazione di proprieta - un processo puo cambiare i diritti che altri domini hanno su un oggetto di sua proprieta. (Proprieta dominio ha oggetto)

Operazione di controllo - un processo puo cambiare i diritti di un dominio di cui il suo dominio ha il controllo. (Dominio controlla un dominio)

Implementazioni possibili poiche e molto grande sta roba

Tabella globale - lista di terne <dominio, risorsa, diritti>.

ACL - per ogni risorsa si memorizza una lista di coppie <dominio, diritti>. Inefficiente su sistemi grandi poiche dev'essere scandita spesso (serve capire se un dominio ha accesso a alla risorsa, e vabbe le scandisci tutte ahia).

CL - capability list. Per ogni dominio si memorizza una lista di coppie <risorsa, diritti> dette capability. La revoca puo essere difficile poiche per togliere il diritto di piu domini a una certa risorsa devi scandire piu capability list.

Meccanismo lock-key - compromesso tra ACL e CL, ogni risorsa ha una lista di stringhe di bit dette lock e ogni dominio ha una lista di stringhe di bit dette key. Un processo in un dominio puo accedere a una risorsa solo se una sua key apre uno dei lock di quella risorsa.

La revoca su ACL e facile. Per revocare i diritti sulle CL pero ci sono varie soluzioni anche se piu difficili (svuotare periodicamente le CL una di loro)

PERIFERICHE

Il sottosistema di IO ha il compito di gestire le periferiche e di permettere ad applicazioni di usarle.

L'**implementazione e a strati**: HW sotttostante, driver per i device che hanno il ruolo di nascondere le differenze tra device dello stesso tipo (modelli diversi di dischi), il resto del sottosistema di IO che esegue operazioni non dipendenti dal dispositivo usando i driver. Il kernel tiene informazioni sulle periferiche in strutture simili alla tabella dei file aperti.

Device differiscono per: velocita di trasferimento, sola scrittura, sola lettura (schermi), bloccanti / non bloccanti, a caratteri / blocchi, condivisibile / dedicato, accesso sequenziale tipo nastri, diretto tipo ssd.

Dispositivi a blocchi - i device di questo tipo devono comprendere i comandi read, write e seek se ad accesso diretto. Dischi principalmente

Dispositivi a caratteri - trasferiscono singoli byte. I device di questo tipo devono comprendere i comandi get e put. Tipi di device cosi sono le tastiere. Si puo usare anche un buffering per leggere righe intere al posto di una sola lettera per volta.

Periferiche di rete - tipicamente i dispositivi di questo tipo sono i socket che permettono a un applicazione di attendere connessioni o di connettersi a un host in attessa per poi comunicare con chiamate simili a read e write.

Orologi e timer - questo tipo di device permettono chiamate per ottenere l'ora, l'uptime del sistema o di fare delle operazioni temporizzate tipo l'esecuzione a un certo orario.

IO bloccanti e non - le chiamate bloccanti bloccano il processo finche non sono completate, e alla fine lo rimettono nello stato di pronto (nastri). Quelle non bloccanti invece non lo bloccano.

Si forniscono dal sottosistema IO

Schedulazione - tipo schedulazione per i dischi dove e utile schedulare la coda delle richiesta in modo da migliorare il tempo di risposta.

Buffering - i buffer sono zone temporanee di memoria che possono essere usate per tenere dati in transito da device a device o da processo a devive e viceversa. Ha la funzione di fare da adattatore tra dispositivi con blocchi di dimensione diversa e di nascondere le differenze di velocita tra 2 dispositivi.

Spooling e prenotazione del device - ci sono dispositivi che non supportano piu di un processo alla volta e quindi non possono essere utilizzati in contemporanea. Il spooler tiene le informazioni per questi tipi di device e li manda in ordine quanto possibile. In alcuni sistemi si puo prenotare l'uso esclusivo di un device.

Gestione errori