

# **Possibili domande di teoria Sistemi Operativi**

(by Akira)

1. *Descrivi i principi di funzionamento del sistema di allocazione noto come "buddy system".*

Il buddy system consiste nel suddividere la memoria in  $2^k$  blocchi di dimensione  $2^L$  (con  $2^L$  eventualmente uguale all'intera memoria). Se un processo richiede una quantità  $s$  di memoria compresa tra  $2^{(L-1)}$  e  $2^L$  allora divido il blocco in due buddy. Ripeto la divisione fino a quando non trovo un blocco "non più divisibile". Posso anche unire due buddy per crearne uno più grande

2. *Mostra i campi dell'Inverted Page Table con una descrizione sintetica.*

La tabella IPT ha 4 campi: Numero di pagina, Id del processo, bit di controllo (che controllano se la pagina trovata è quella giusta oppure no), campo di concatenazione (rimanda ad un'altra riga fino a che non si trova quella cercata).

3. *Quale è l'algoritmo usato dal processore per ottenere la corretta page table entry?*

Controlla la TLB, se la trova la CPU genera l'indirizzo fisico, altrimenti accede alla PT, se la trova in memoria principale, aggiorna la TLB e la CPU genera l'indirizzo fisico. Altrimenti comincia la subroutine Page Fault Handling ovvero il SO da istruzioni alla CPU per leggere la pagina dal disco, poi attiva I/O HW, la pagina trovata viene trasferita in memoria principale, e se la memoria non è piena viene aggiornata la PT. Se è piena viene effettuato un page replacement e conseguentemente viene aggiornata la PT

4. *Supponi di identificare ciascun blocco con l'indirizzo iniziale e con un numero  $n$  tale che la taglia del blocco sia  $2^n$ . Dai una regola per stabilire se due blocchi sono buddies.*

Due blocchi sono buddies se è possibile effettuare un merge fra i due. Prima di tutto devono avere la stessa dimensione ed essere contigui (DA FINIRE)

5. *Perché la tecnica **page fault frequency** può essere considerata una approssimazione dell'approccio working set?*

Perché nel PFF si fissa un valore ottimale  $f$  della frequenza di page fault che si assume essere corrispondente alla situazione  $|RS|=|WS|$ , per regolare la dimensione del RS

6. *Che succede in page fault frequency nella **transizione tra due zone di località** che accedono a pagine distinte?*

In questa fase avvengono numerosi page fault, che portano il processo ad aumentare il proprio RS a discapito degli altri processi, che possono essere sospesi o trasferiti su disco

7. Descrivi **sinteticamente** le differenze tra un sistema operativo basato sul modello "single address space" (SAS) e uno basato sul modello "multiple address space" (MAS)?

In SAS esiste un singolo spazio di indirizzamento, il che semplifica lo scambio di dati fra processi. Nei sistemi SAS vengono utilizzate tipicamente le IPT. Nei sistemi MAS ciascun processo ha un proprio spazio di indirizzamento.

8. Che significa page buffering?

Il page buffering è un sistema di replacement delle pagine in memoria virtuale. Consiste nel salvare le pagine che devono essere sostituite in un buffer, per migliorare le prestazioni.

9. Che significa "minor page fault" in questo contesto? Elenca i vantaggi di page buffering per la gestione delle pagine modificate.

Un minor page fault avviene quando la pagina da cercare è nel page buffer. In questo caso il page fault non provoca accesso al disco. Ci sono anche vantaggi in scrittura, in quanto la scrittura può essere rimandata se il frame non deve essere usato immediatamente ed eseguita dal disco come attività a bassa priorità, questo perché le pagine modificate quando vengono tolte dal resident set devono essere scritte sul disco.

10. Descrivi l'algoritmo di **disk scheduling** "elevator". Perché viene introdotta la tecnica del "request merging" e quali sono i vantaggi?

L'algoritmo elevator consiste nel servire le richieste man mano che la testina si muove nella stessa direzione. Quando la testina arriva alla fine (o non ha più richieste nella stessa direzione) torna indietro e serve tutte le richieste che incontra tornando. La tecnica request merging consiste nel considerare tutte le richieste contigue come se fossero un'unica richiesta. Questa tecnica permette di risparmiare in termini di overhead.

11. Spiega il **write-starving-reads problem** nell'ambito dello scheduling del disco e descrivi la tecnica usata dallo scheduling "anticipatory" per affrontare tale problema.

Le scritture non sono bloccanti mentre le letture lo sono. Questo causa l'arrivo di tutte le richieste di scrittura contemporaneamente, mentre per continuare la lettura devo attendere i risultati. Quindi vengono favorite le scritture causando *unfairness*. Anticipatory "guarda avanti", ovvero aspetta per 6ms un'altra lettura, perché tipicamente è vicina. Questa

tecnica riduce il problema write-starving-problem perché limita "il potere" delle scritture.

12. Dai la definizione di Working Set per un processo specificando il significato del parametro delta  $\Delta$ .

Il working set è l'insieme di indirizzi di memoria referenziati che usa un processo di cardinalità pari a  $\Delta$ . La situazione ideale è che  $|WS| = |RS|$

13. Confronta la politica "resident set=working set" con la politica LRU. Analogie e differenze.

$|RS|=|WS|$  consiste nell'eliminare le pagine dal resident set che non si trovano nel working set, il che coincide con LRU. In più rispetto all'LRU questa strategia permette di eseguire un processo solo se il suo working set è in memoria principale, e inoltre si ha un monitoraggio del WS di ogni processo.

14. Mostra lo schema di segmentazione paginata.

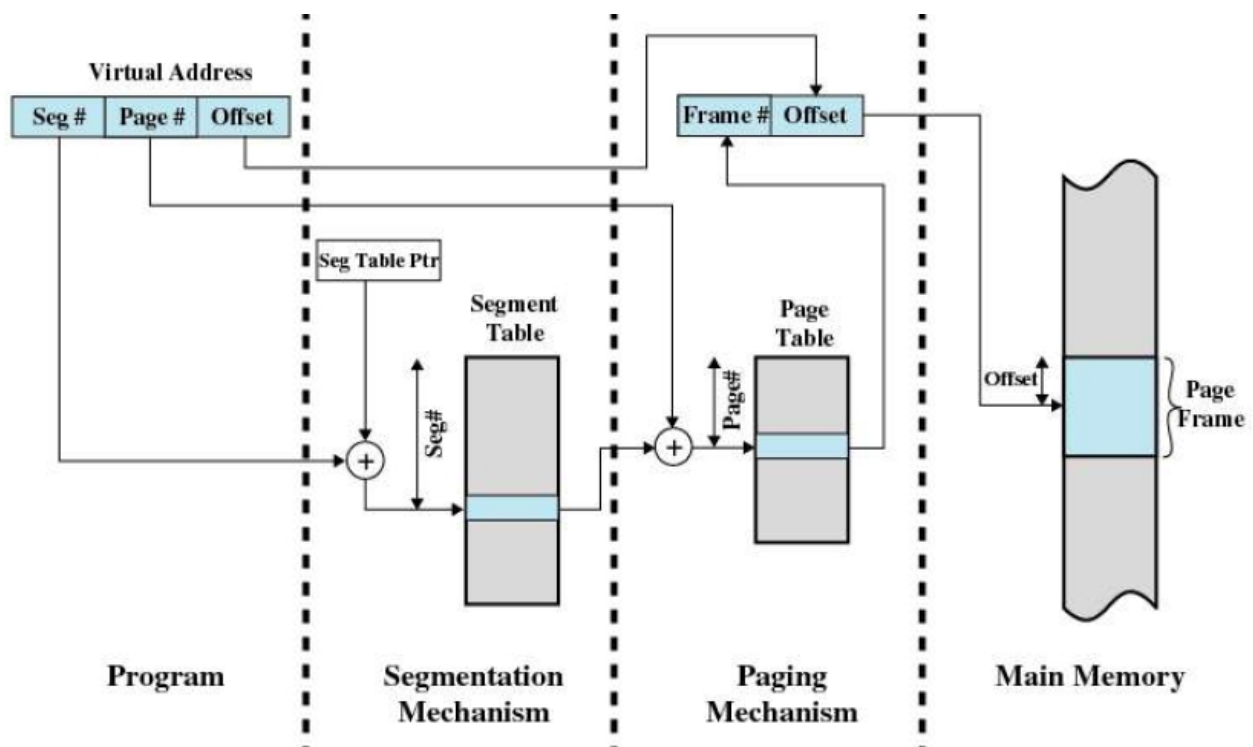


Figure 8.13 Address Translation in a Segmentation/Paging System

15. Illustra il ruolo degli interrupt nell'ambito delle varie fasi della gestione di un page fault.

Un interrupt fa scattare un mode switch per permettere al dispatcher di mandare il blocco il processo che ha causato il page fault e mandare in running un processo presente nella coda ready. Alla fine della gestione del page fault, un interrupt avvisa il dispatcher che il processo che l'aveva causata è di nuovo ready.

*16. Descrivi una tecnica usata nella gestione della memoria virtuale che puo considerarsi una approssimazione dell'approccio working set e spiegate il motivo.*

Il Page Fault Frequency è una approssimazione dell'approccio WS perché la  $|WS| \neq |RS|$ . PFF a seconda dell'aumento o diminuzione dei page fault aumenta o decrementa il delta associato al WS. In questo quindi si approssima all'approccio WS proprio perché tende a rendere uguali  $|WS|$  e  $|RS|$ .

*17. Cosa contiene l'inode nel filesystem di un sistema operativo UNIX? come è correlato l'inode al concetto di nome di file e di directory?*

L'inode contiene informazioni sul file (data di creazione, permessi etc.) e i riferimenti ai blocchi fisici del file. Il nome di un file è l'hard link. Un inode può avere diversi hard link (quindi diversi nomi), ma un inode descriverà sempre solo un file fisico. Una directory contiene una lista di coppie di inode-nome file.

*18. Quale è il modo migliore di sistemare i blocchi del file sul disco?*

Il modo migliore di sistemare i blocchi è in settori consecutivi all'interno di una stessa traccia, e in tracce consecutive nel senso di movimento della testina durante la lettura.

*19. in particolare quale è il modo migliore di sistemare i blocchi dei file che occupano più di una traccia?*

In tracce consecutive.

*20. Confronta l'architettura "tabella delle pagine a più livelli" con l'architettura "inverted page table" e mostra quali sono i pro e i contro dei due approcci.*

L'architettura "tabella delle pagine a più livelli" permette di non avere in memoria l'intera tabella, indirizzando nel primo livello le pagine del secondo e così via fino ad arrivare al livello fisico. Rispetto all'inverted page si hanno molti page fault e un doppio accesso alla memoria. L'inverted page table permette di avere in memoria l'intera tabella costruita attraverso una funzione di hash. Ma per la gestione delle collisioni il numero di accessi alla memoria non è costante.

21. Descrivi l'architettura dei sistemi operativi "process-based" (anche detta microkernel), descrivendo vantaggi e svantaggi rispetto a quella "execution within process" (stile UNIX)

Nei "process based" tutto compreso il sistema operativo viene eseguito come un processo, quindi il sistema è modulare e quindi "elegante" e funzionale ma è difficile da programmare. Nell'"execution within process" parte del kernel è eseguito insieme al processo, rendendo più efficienti gli scambi di informazioni tra kernel e i processi.

22. Descrivi l'architettura hw per memoria virtuale con tabella delle pagine a due livelli. Spiega anche in che senso tale architettura risolve il problema delle tabelle delle pagine molto grandi.

23. Che cosa è una "system call"? Elenca le categorie di funzionalità che ritieni più importanti tra quelle fornite dalle system call.

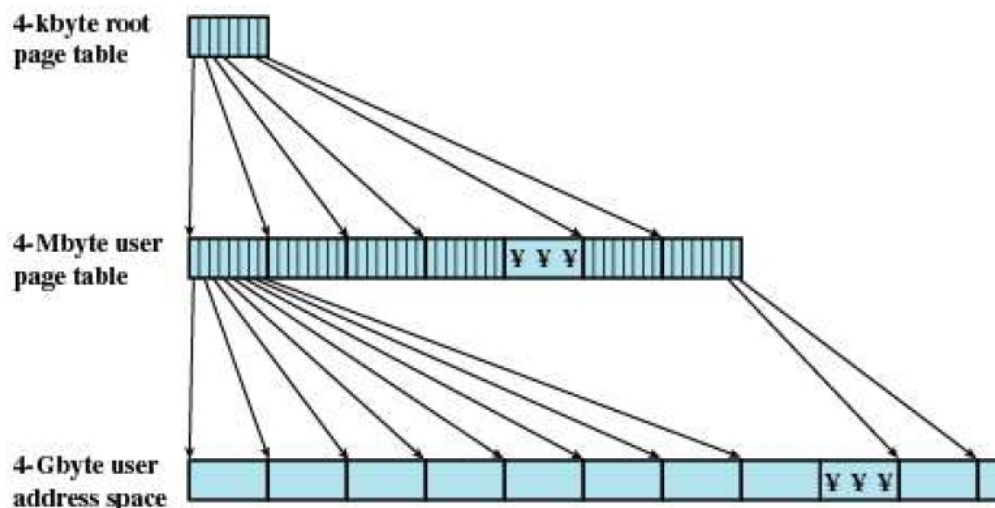
E' una operazione eseguibile solamente dal kernel. Le categorie di funzionalità più importanti sono l'I/O, il lancio di nuovi processi, richiesta di nuova memoria e interrupt handler.

24. Che cosa è un mode switch? In quali occasioni si verifica?

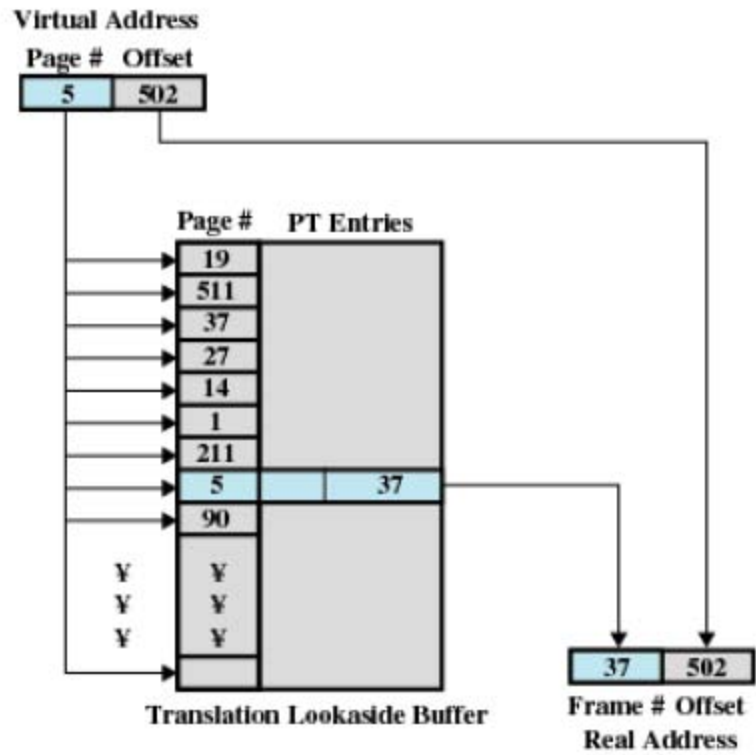
Un mode switch è il passaggio da user mode a kernel mode o viceversa. Si verifica quando un processo richiede l'I/O, oppure quando il quanto di tempo è scaduto, o quando si verifica una interrupt.

25. Descrivi il concetto di tabella delle pagine a più livelli e mostra i dati contenuti nelle singole righe delle tabelle.

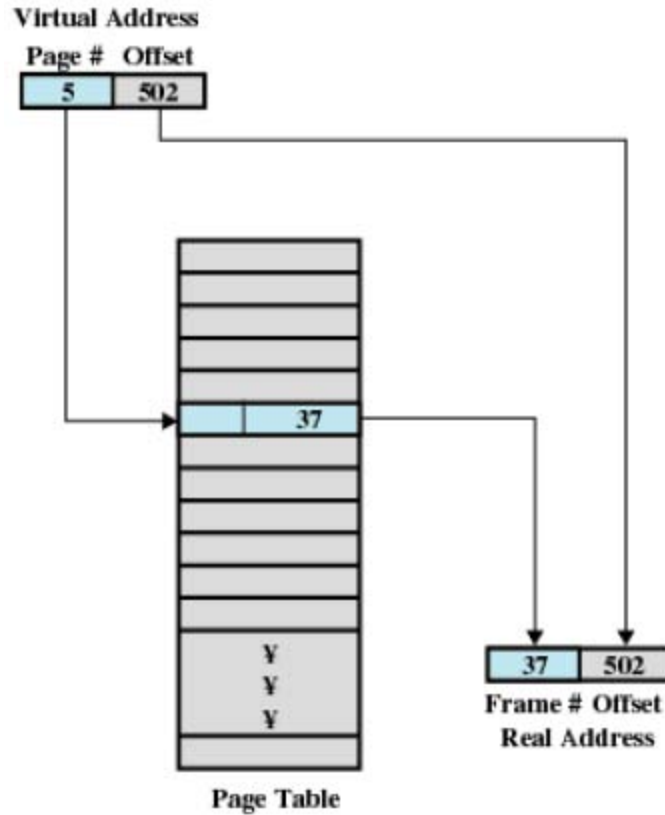
In ogni livello ho i riferimenti al successivo livello fino a raggiungere quello fisico.



*26. Mostra lo schema dell'architettura del TLB. Che vantaggi dà l'adozione del TLB? Pensi che il TLB sia più utile in uno schema di paginazione a 1 livello o a 4 livelli?*



(b) Associative mapping

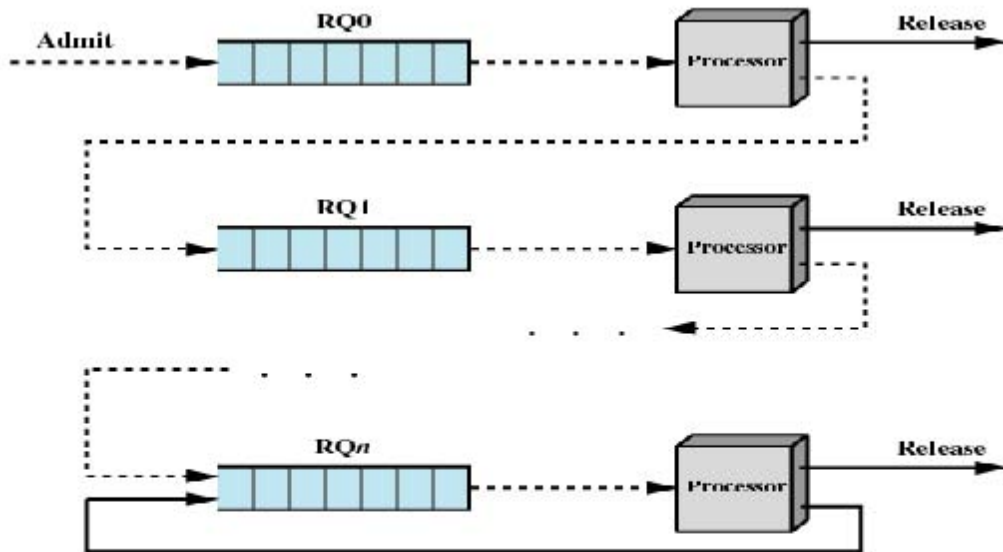


(a) Direct mapping

Serve a diminuire il numero di accessi in memoria principale. Contiene le pagine usate più di recente, ed effettua un mapping tra il numero di pagina e la entry nella page table. Con un livello ho una TLB molto grande.

27. *Descrivi la tecnica di scheduling denominata feedback.*

E' preemptive, usa FCFS in tutte le code che sono fatte in base alla durata dell'utilizzo del processore e RR nell'ultima con priorità più bassa, e ha coda di priorità.



**Figure 9.10 Feedback Scheduling**

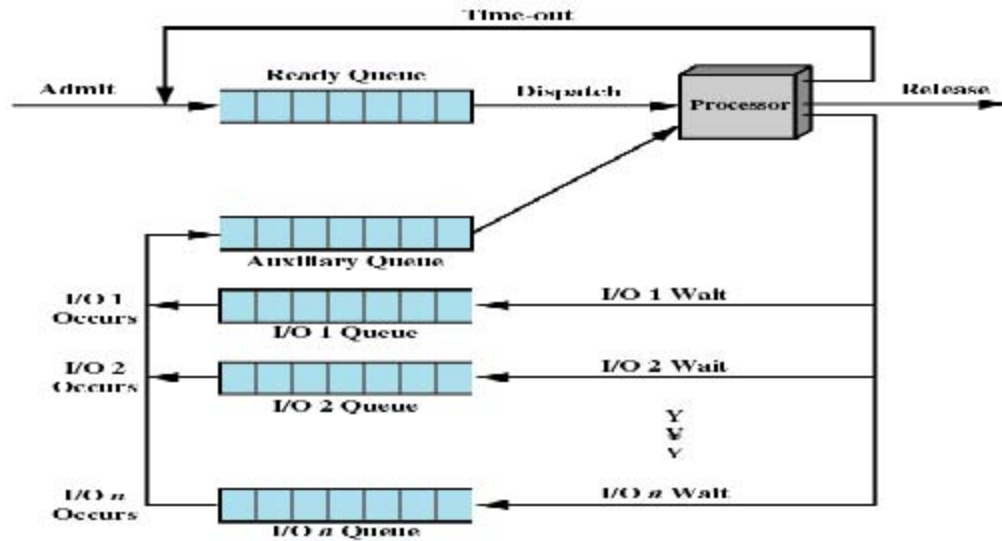
28. *Analogie e differenze tra le tecniche di memoria virtuale e disk caching.*

Analogie: entrambe le tecniche mirano a tenere in memoria solo dati e programmi acceduti di frequente.

Differenze: le tecniche agiscono in domini differenti: virtual memory (processi, pagine, segmenti); disk caching (files).

29. *In quali casi Round Robin è unfair? Mostra uno schema che lo rende fair.*

E' unfair perchè favorisce i processi CPU bound rispetto a quelli I/O bound in quanto se quest'ultimi vengono interrotti prima del loro quanto di tempo a disposizione vanno comunque in blocco. Lo schema che lo risolve è quello del VRR ovvero mettendo in una coda con maggiore priorità quei processi che sono stati interrotti prima che il loro quanto di tempo sia scaduto. Questi processi opereranno per il tempo rimanente che gli era rimasto prima che venissero interrotti.

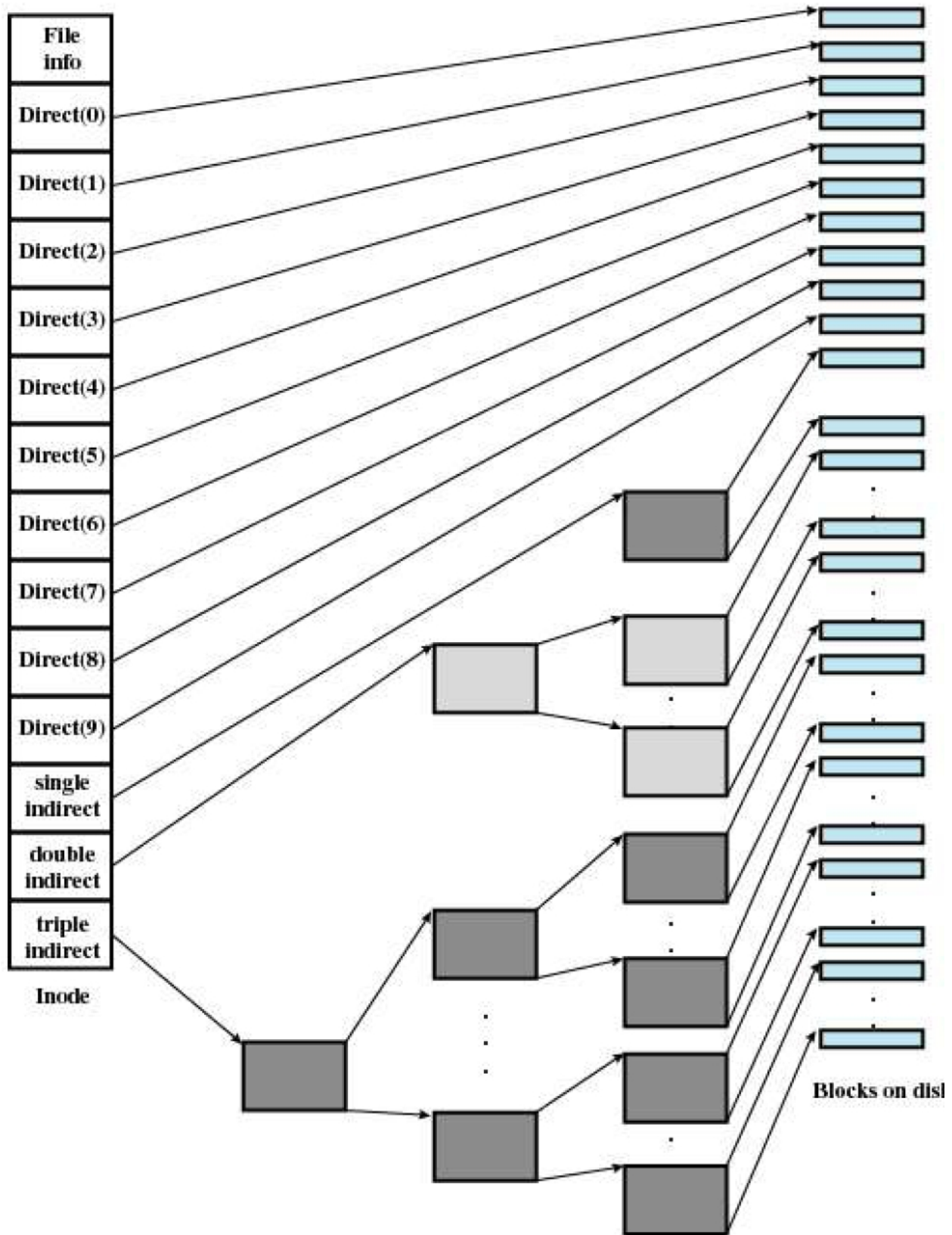


**Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler**

30. *Elenca sinteticamente il contenuto della struttura Inode nei sistemi UNIX.*

Un inode descrive vari attributi di un file (utente, gruppo, permessi, data/ora di modifica e accesso, ecc) e quali blocchi del disco contengono il file.

31. *Mostra la struttura gerarchica che i Filesystem dei sistemi UNIX utilizzano per tenere traccia dei blocchi di un file e che ha la sua radice nell' Inode.*

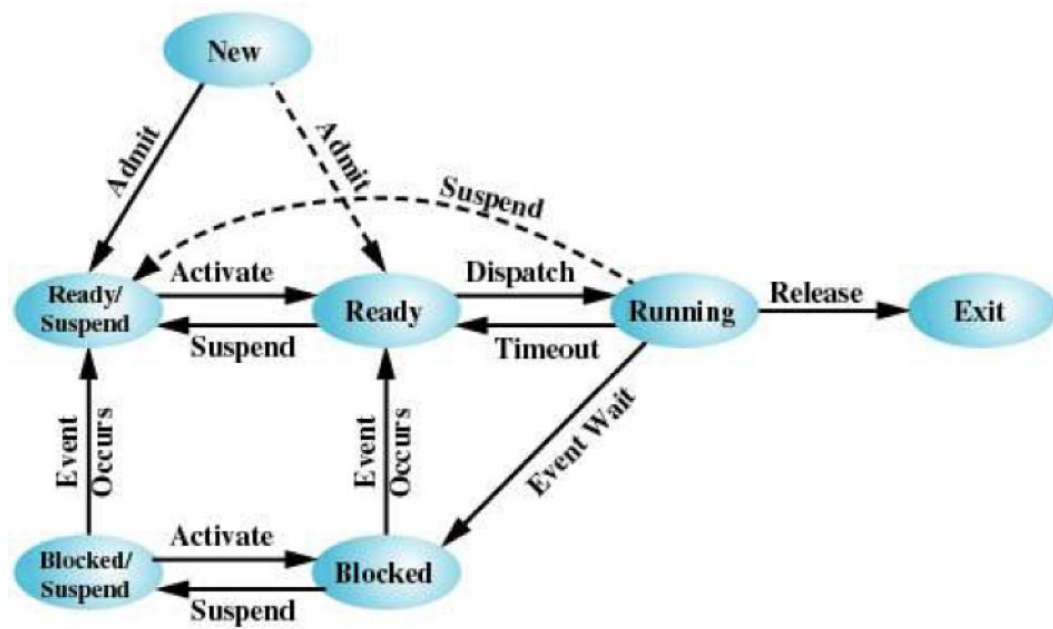


**Figure 12.13** Layout of a UNIX File on Disk

32. Descrivi l'architettura hw per memoria virtuale con tabella delle pagine invertita. Spiega anche in che senso permette di evitare di avere tabelle delle pagine molto grandi.

Nella IPT si ha un'unica page table, costituita da 4 campi: numero di pagina, id processo, control bits, chain. Alla table table si accede tramite una funzione hash. Si evitano tabelle grandi grazie all'uso dell'hash e della concatenazione.

33. Mostra il diagramma di stato di un processo in cui viene considerata la possibilità di sospensione del processo stesso (quello a 7 stati). Perché un processo dovrebbe essere sospeso? quando può essere riattivato?



(b) With Two Suspend States

Un processo viene sospeso quando richiede operazioni di I/O con una conseguente liberazione di memoria. Viene riattivato quando finisce la sua operazione.

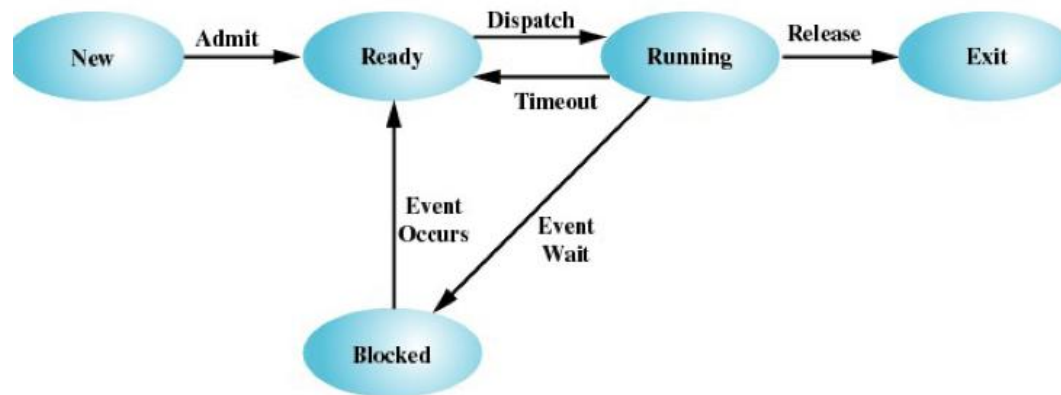
34. Devi progettare lo scheduling di un sistema in cui sono presenti processi che interagiscono con l'utente e decidi per un approccio round robin. In base a quali considerazioni decidi la durata del quanto di tempo?

Le considerazioni sono sul fatto se i CPU burst sono o meno trascurabili in quanto RR favorisce comunque i processi CPU bound. Se non ho processi CPU bound allora in coda Ready i processi non attendono nulla. Decido la durata del quanto di tempo in base alla grandezza minima del burst dei processi.

35. Descrivi le funzionalità del dispatcher. In quali occasioni viene eseguito?

Il Dispatcher sceglie i processi da eseguire dalla coda dei Ready e inoltre gestisce anche il fatto di metter in blocco quei processi che sono in attesa di un evento. Ha differenti politiche dettate dallo scheduling come ad esempio preemptive o round-robin.

36. Mostra il diagramma di transizione di stato di un processo per il modello a 5 stati (3+new+exit). Descrivi ciascuno dei 3 stati principali.



Ready: I processi in coda in attesa di esser eguiti

Running: I processi in esecuzione

Blocked: I processi che erano in esecuzione e sono stati bloccati in attesa di un evento, quando questo evento avviene il dispatcher li mette nella coda dei Ready.

37. Descrivi brevemente l'algoritmo di I/O scheduling C-SCAN. Dai un esempio di starvation o unfairness.

C-SCAN è una variante dell'algoritmo ELEVATOR. Quando la testina arriva in fondo dopo aver servito le richieste incontrate, torna all'inizio. E' fair in quanto tornando all'inizio serve quelle appena entrate in coda senza farle aspettare troppo. Dà starvation in quanto potrebbe andare avanti all'infinito se continua ad avere richieste nella stessa direzione, senza quindi tornare indietro a servire quelle precedenti.