Di un sistema è noto che la tabella delle pagine più grande del sistema occupa esattamente 2 frame, il numero di un frame è scritto su 4 byte usando solo i primi 26 bit, e nel sistema sono presenti in media 4 processi che insieme producono una frammentazione interna complessiva media di 8 Kilobyte.

lo spazio logico del sistema è grande: 8 Megabyte ✔

lo spazio fisico del sistema è grande: 256 Gigabyte ✔

16 Megabyte	non si può ricavare dai dati del problema	nessuno dei valori proposti	512 Gigabyte
1 Megabyte			4 Gigabyte

Risposta corretta.

La risposta corretta è:

Di un sistema è noto che la tabella delle pagine più grande del sistema occupa esattamente 2 frame, il numero di un frame è scritto su 4 byte usando solo i primi 26 bit, e nel sistema sono presenti in media 4 processi che insieme producono una frammentazione interna complessiva media di 8 Kilobyte.

lo spazio logico del sistema è grande: [8 Megabyte]
lo spazio fisico del sistema è grande: [256 Gigabyte]

Dopo l'esecuzione dei seguenti comandi in un ambiente Unix (come visti a lezione):

1: cd /tmp
2: mkdir newfolder
3: cd newfolder
4: echo "ciao" > pippo // crea un nuovo file di nome *pippo* contenente la stringa *ciao*
5: ln pippo paperino
6: ln ../newfolder folder2
7: cp paperino topolino
8: echo "salve" >> topolino // aggiunge "salve" a fondo file
9: rm pippo
10: cat paperino // *cat* stampa il contenuto del file passato come argomento
11: mkdir ../folder3

Scegli un'alternativa:

a.
1. il link-counter dell'i-node di *paperino* è: 2
2. il link counter di *tmp* è: aumentato di 2
3. l'output del comando 10 è: "ciao"
4. il comando 6 da come risultato: un nuovo collegamento alla cartella newfolder

b. ✔
1. il link-counter dell'i-node di *paperino* è: 1
2. il link counter di *tmp* è: aumentato di 2
3. l'output del comando 10 è: "ciao"
4. il comando 6 da come risultato: un errore perché non sono ammessi hard link tra cartelle

c.
1. il link-counter dell'i-node di *paperino* è: 1
2. il link counter di *tmp* è: 2
3. l'output del comando 10 è: no such file or directory
4. il comando 6 da come risultato: un errore perché non sono ammessi hard link tra cartelle

d.
1. il link-counter dell'i-node di *paperino* è: 2
2. il link counter di *tmp* è: aumentato di 1
3. l'output del comando 10 è: "ciao" seguito da "salve"
4. il comando 6 da come risultato: un errore perché non sono ammessi hard link tra cartelle

Risposta corretta.

La risposta corretta è:

1. il link-counter dell'i-node di *paperino* è: 1
2. il link counter di *tmp* è: aumentato di 2
3. l'output del comando 10 è: "ciao"
4. il comando 6 da come risultato: un errore perché non sono ammessi hard link tra cartelle

In hard disk grande 512 Gigabyte, per scrivere il numero di un blocco vengono usati 28 bit, arrotondati al minimo numero di byte necessario. L'hard disk adotta una allocazione indicizzata semplice, e di un file A si sa che nel suo blocco indice 16 byte vengono usati per tenere traccia dei blocchi di dati di A. Quanto può essere grande al massimo A?

Scegli un'alternativa:

- a. 16 Kilobyte ✖
- b. 20 Kilobyte
- c. 12 Kilobyte
- d. 8 Kilobyte

Risposta errata.

La risposta corretta è: 8 Kilobyte

All'interno di un sistema operativo che implementa la memoria virtuale, un certo processo P è correntemente in stato di "*Waiting for page*", e si sa che, una volta ritornato in stato "*running*" non dovrà più rilasciare la CPU volontariamente prima di aver terminato la propria esecuzione (in altre parole, non deve più eseguire operazioni di I/O, di sincronizzazione o di comunicazione con altri processi, e non genererà più alcun page fault).

Quale delle seguenti coppie di algoritmi di scheduling garantisce che il processo P riuscirà a portare a termine la propria computazione, e perché? (si assuma che SJF possa effettivamente essere implementato)

Scegli un'alternativa:

- a. poiché una volta servito il page fault il processo torna running, solo RR e SJF non preemptive garantiranno che il processo possa completare la sua esecuzione senza venire più interrotto.
- b. poiché una volta servito il page fault il processo viene rimesso in coda di ready, solo FCFS e SJF non preemptive garantiranno che il processo, una volta tornato running, possa completare la sua esecuzione senza venire più interrotto. ✖
- c. poiché una volta servito il page fault il processo torna running, solo FCFS e SJF non preemptive garantiranno che il processo possa completare la sua esecuzione senza venire più interrotto.
- d. poiché una volta servito il page fault il processo tornerà in coda di ready, solo FCFS e RR garantiranno la terminazione del processo.

Risposta errata.

La risposta corretta è: poiché una volta servito il page fault il processo tornerà in coda di ready, solo FCFS e RR garantiranno la terminazione del processo.

si consideri l'esecuzione della seguente porzione di codice che utilizza la system call fork:

```
int a, b, c, d, n, pid1, pid2, pid3;
a = 30, b = 40, c = 50, d = 60;
n = fork();
      if ( n == 0)
         {a = 35; b = 45;
          pid1 = getppid();
          printf("%d", pid1);
          exit(0);}
      else
         {c = 55; d = 65;
          pid2 = getpid();
          printf("%d",pid2);
          pid3 = wait(NULL);
          exit(0);}
```

il valore della variabile a vista dal processo figlio subito prima della sua exit è [35] ✔

il valore della variabile c vista dal processo figlio subito prima della sua exit è [50] ✔

il valore della variabile b vista dal processo padre subito prima della sua exit è: [40] ✔

il valore della variabile d vista dal processo padre subito prima della sua exit è: [65] ✔

all'esecuzione delle due printf vale la seguente relazione: pid1 [=] ✔ pid2

del risultato della wait possiamo dire che: pid1 [>] ✖ pid3

[60] [45] [=] [50] [40] [35] [30] [55] [65] [<] [>]

Risposta parzialmente esatta.

Hai selezionato correttamente 5.

La risposta corretta è:

si consideri l'esecuzione della seguente porzione di codice che utilizza la system call fork:

```
int a, b, c, d, n, pid1, pid2, pid3;
a = 30, b = 40, c = 50, d = 60;
n = fork();
      if ( n == 0)
         {a = 35; b = 45;
          pid1 = getppid();
          printf("%d", pid1);
          exit(0);}
      else
         {c = 55; d = 65;
          pid2 = getpid();
          printf("%d",pid2);
          pid3 = wait(NULL);
          exit(0);}
```

il valore della variabile a vista dal processo figlio subito prima della sua exit è [35]
il valore della variabile c vista dal processo figlio subito prima della sua exit è [50]
il valore della variabile b vista dal processo padre subito prima della sua exit è: [40]
il valore della variabile d vista dal processo padre subito prima della sua exit è: [65]
all'esecuzione delle due printf vale la seguente relazione: pid1 [=] pid2
del risultato della wait possiamo dire che: pid1 [<] pid3

Tre processi P1 P2 e P3 arrivano uno dopo l'altro – nell'ordine indicato ma in tempi diversi – in coda di ready. I tre processi sono caratterizzati da un unico burst di CPU e da nessun burst di I/O. In quale caso il waiting time medio del sistema relativo ai tre processi risulta lo stesso sia adottando un algoritmo di scheduling FCFS che adottando una algoritmo di scheduling SJF preemtpive?

Scegli un'alternativa:

- a. è sufficiente che: Burst_P1 > Burst_P2 > Burst_P3
- b. è necessario che: Burst_P1 << Burst_P2 << Burst_P3
- c. mai, perché SJF fornisce un waiting time medio migliore di FCFS in qualsiasi caso ✖
- d. è sufficiente che: Burst_P1 ≤ Burst_P2 ≤ Burst_P3

Risposta errata.

La risposta corretta è: è sufficiente che: Burst_P1 ≤ Burst_P2 ≤ Burst_P3

Nei sistemi operativi moderni il turnaround di un processo può variare moltissimo da una esecuzione alla successiva. A che cosa è dovuto questo comportamento?

Scegli un'alternativa:

a. All'uso di librerie statiche, che vengono caricate in RAM solo se usate in una certa esecuzione, influenzanto così il turnaround a seconda che vengano riferite o no

b. Alla presenza della memoria paginata, che fa si che l'effettivo tempo di turnaround dipenda fortemente dalla porzione di PT che può essere memorizzata nel TLB in una certa esecuzione ✖

c. Alla presenza della memoria virtuale, che fa si che l'effettivo tempo di turnaround dipenda fortemente dal numero di page fault generati dal processo in una certa esecuzione

d. Al fatto che venga adottata una paginazione semplice o una a più livelli. Infatti nella paginazione a più livelli il costo della traduzione degli indirizzi aumenta enormemente i tempi di esecuzione

Risposta errata.

La risposta corretta è: Alla presenza della memoria virtuale, che fa si che l'effettivo tempo di turnaround dipenda fortemente dal numero di page fault generati dal processo in una certa esecuzione

Ricostruite il codice del generico lettore nel problema dei lettori-scrittori:

```
semaphore mutex = 1, scrivi = 1;
int numlettori= 0;
Processo lettore {
wait(mutex);
    numlettori++;        ✔
    if numlettori == 1 wait(scrivi);   ✔
  signal(mutex);
  ... leggi il file ...
  wait(mutex);
    numlettori--;        ✔
    if numlettori == 0 signal(scrivi);   ✔
  signal(mutex)
}
```

if numlettori >= 1 wait(scrivi); if numlettori == 0 wait(scrivi); if numlettori > 0 wait(scrivi); if numlettori <= 1 wait(scrivi); if numlettori == 1 signal(scrivi);

Su un hard disk che adotta una allocazione concatenata (senza FAT) è memorizzato un file A della dimensione di 0x4000 byte, e si sa che nell'ultimo blocco di A sono presenti 16 byte del file. Si sa inoltre che per scrivere il numero di un blocco vengono usati 27 bit, arrotondati al minimo numero di byte necessario. Quanto è grosso l'hard disk?

Scegli un'alternativa:

a. 128 Gigabyte

b. 512 Gigabyte

c. 1 Terabyte

d. 256 Gigabyte ✖

Risposta errata.

La risposta corretta è: 512 Gigabyte

Le caratteristiche fondamentali dell'algoritmo di scheduling RR sono:

Scegli un'alternativa:

a.
1. al diminuire del quanto di tempo aumenta l'overhead dei context switch
2. fornisce mediamente un buon tempo di risposta
3. al diminuire del quanto di tempo tende a comportarsi come FCFS
4. è l'algoritmo di base naturale dei sistemi time sharing

b.
1. al diminuire del quanto di tempo aumenta l'overhead dei context switch
2. fornisce mediamente waiting time migliori di quelli di SJF
3. all'aumentare del quanto di tempo tende a comportarsi come FCFS
4. è l'algoritmo di base naturale dei sistemi time sharing

c. ✔
1. fornisce mediamente un buon tempo di risposta
2. è l'algoritmo di base naturale dei sistemi time sharing
3. al diminuire del quanto di tempo aumenta l'overhead dei context switch
4. all'aumentare del quanto di tempo tende a comportarsi come FCFS

d.
1. all'aumentare del quanto di tempo tende a comportarsi come FCFS
2. è l'algoritmo di base naturale dei sistemi time sharing
3. al diminuire del quanto di tempo diminuisce l'overhead dei context switch
4. fornisce mediamente un buon tempo di risposta

Risposta corretta.

La risposta corretta è:

1. fornisce mediamente un buon tempo di risposta
2. è l'algoritmo di base naturale dei sistemi time sharing
3. al diminuire del quanto di tempo aumenta l'overhead dei context switch
4. all'aumentare del quanto di tempo tende a comportarsi come FCFS

Un sistema operativo deve adottare una paginazione a due o più livelli anziché una paginazione semplice quando:

Scegli un'alternativa:

a. quando la tabella delle pagine più grande del sistema ha una dimensione maggiore di un frame, perché il SO potrebbe non riuscire a trovare due frame adiacenti in cui allocare la PT ✔

b. quando la tabella delle pagine più grande del sistema ha una dimensione maggiore di un frame, perché in questo caso la traduzione degli indirizzi da logici a fisici risulta più efficiente

c. quando il sistema ha uno spazio di indirizzamento fisico molto grande, perché in questo caso le tabelle delle pagine tendono ad essere sono molto grandi, e una paginazione a più livelli è più facilmente gestibile dal SO

d. quando l'architettura non mette a disposizione un TLB, perché in questo caso la paginazione a due o più livelli rende la traduzione degli indirizzi da logici a fisici più efficiente

Risposta corretta.

La risposta corretta è: quando la tabella delle pagine più grande del sistema ha una dimensione maggiore di un frame, perché il SO potrebbe non riuscire a trovare due frame adiacenti in cui allocare la PT