In un moderno sistema time-sharing con uno spazio di indirizzamento logico maggiore di quello fisico, per quali ragioni un processo può dover transire dallo stato "running" allo stato "waiting"?

Scegli un'alternativa:

a.
1. Perché ha completato una operazione di I/O
2. Perché si è addormentato su un semaforo
3. Perché ha indirizzato una pagina non presente in RAM (page fault)

b.
1. Perché deve compiere una operazione di I/O
2. Perché si è addormentato su un semaforo
3. Perché ha indirizzato una pagina vittima (page fault)

c. ✔
1. Perché deve compiere una operazione di I/O
2. Perché si è addormentato su un semaforo
3. Perché ha indirizzato una pagina non presente in RAM (page fault)

d.
1. Perché deve compiere una operazione di I/O
2. Perché è stato svegliato su un semaforo
3. Perché ha indirizzato una pagina non presente in RAM (page fault)

Risposta corretta.

La risposta corretta è:

1. Perché deve compiere una operazione di I/O
2. Perché si è addormentato su un semaforo
3. Perché ha indirizzato una pagina non presente in RAM (page fault)

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 = 60, b = 70, c = 80, d = 90;
n = fork();
      if ( n == 0)
         {a = 65; b = 75;
          pid1 = getppid();
          printf("%d", pid1);
          exit(0);}
      else
         {c = 85; d = 95;
          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 [60] ✖

il valore della variabile c vista dal processo figlio subito prima della sua exit è [85] ✖

il valore della variabile b vista dal processo padre subito prima della sua exit è: [75] ✖

il valore della variabile d vista dal processo padre subito prima della sua exit è: [90] ✖

all'esecuzione delle due printf vale la seguente relazione: pid1 [>] ✖ pid2

del risultato della wait possiamo dire che: pid2 [<] ✔ pid3

[60] [>] [75] [<] [95] [70] [65] [80] [90] [=] [85]

Risposta parzialmente esatta.

Hai selezionato correttamente 1.

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 = 60, b = 70, c = 80, d = 90;
n = fork();
      if ( n == 0)
         {a = 65; b = 75;
          pid1 = getppid();
          printf("%d", pid1);
          exit(0);}
      else
         {c = 85; d = 95;
          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 [65]
il valore della variabile c vista dal processo figlio subito prima della sua exit è [80]
il valore della variabile b vista dal processo padre subito prima della sua exit è: [70]
il valore della variabile d vista dal processo padre subito prima della sua exit è: [95]
all'esecuzione delle due printf vale la seguente relazione: pid1 [=] pid2
del risultato della wait possiamo dire che: pid2 [<] pid3

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 -s /tmp/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: 1
2. il link counter di *newfolder* è: 2
3. l'output del comando 10 è: "ciao"
4. il link counter di tmp è: aumentato di 3

b.
1. il link-counter dell'i-node di *paperino* è: 1 ✔
2. il link counter di *newfolder* è: 2
3. l'output del comando 10 è: "ciao"
4. il link counter di tmp è: aumentato di 2

c.
1. il link-counter dell'i-node di *paperino* è: 2
2. il link counter di *newfolder* è: 2
3. l'output del comando 10 è: "ciao"
4. il link counter di tmp è: aumentato di 3

d.
1. il link-counter dell'i-node di *paperino* è: 2
2. il link counter di *newfolder* è: 2
3. l'output del comando 10 è: "ciao"
4. il link counter di tmp è: aumentato di 1

Risposta corretta.

La risposta corretta è:

1. il link-counter dell'i-node di *paperino* è: 1
2. il link counter di *newfolder* è: 2
3. l'output del comando 10 è: "ciao"
4. il link counter di tmp è: aumentato di 2

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 FCFS 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 RR 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.

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

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

if numlettori > 0 wait(scrivi);

if numlettori == 0 wait(scrivi);

if numlettori == 1 wait(scrivi);

if numlettori >= 1 wait(scrivi);

if numlettori == 0 signal(scrivi);

Risposta errata.

La risposta corretta è:
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)
}
```

Un sistema operativo moderno adotta un algoritmo di sostituzione delle pagine per:

Scegli un'alternativa:

a. stabilire quale, tra le pagine dei processi utente correntemente in RAM possa essere rimossa per fare spazio ad una pagina mancante che è stata indirizzata da un processo in esecuzione. E' auspicabile che la pagina scelta sia una pagina che non serve più al processo a cui appartiene, o almeno che non sia stata riferita di recente. ✔

b. stabilire quale, tra le pagine dei processi utente correntemente in RAM possa essere rimossa per fare spazio ad una pagina mancante che è stata indirizzata da un processo in esecuzione. E' obbligatorio che la pagina scelta sia una pagina che non serve più al processo a cui appartiene, e che non sia stata riferita di recente.

c. stabilire quale, tra le pagine dei processi utente correntemente in RAM possa essere rimossa per fare spazio ad una pagina più importante che è stata indirizzata da un processo in esecuzione. E' auspicabile che la pagina scelta sia una pagina che non serve più al processo a cui appartiene, o almeno che sia stata riferita di recente.

d. stabilire quale, tra le pagine dei processi utente correntemente in RAM possa essere rimossa per fare spazio ad una pagina più importante che è stata indirizzata da un processo in esecuzione. E' auspicabile che la pagina scelta sia una pagina che non serve più al processo a cui appartiene, o almeno che non sia stata ancora utilizzata.

Risposta corretta.

La risposta corretta è: stabilire quale, tra le pagine dei processi utente correntemente in RAM possa essere rimossa per fare spazio ad una pagina mancante che è stata indirizzata da un processo in esecuzione. E' auspicabile che la pagina scelta sia una pagina che non serve più al processo a cui appartiene, o almeno che non sia stata riferita di recente.

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 15 bit, arrotondati al minimo numero di byte necessario. Quanto è grosso l'hard disk?

Scegli un'alternativa:

a. 64 Megabyte ✔

b. 32 Megabyte

c. 128 Megabyte

d. 256 Megabyte

Risposta corretta.

La risposta corretta è: 64 Megabyte

In hard disk grande 512 Gigabyte, per scrivere il numero di un blocco vengono usati 25 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 12 byte vengono usati per tenere traccia dei blocchi di dati di A. Quanto può essere grande al massimo A?

Scegli un'alternativa:

- a. 48 Kilobyte ✔
- b. 32 Kilobyte
- c. 24 Kilobyte
- d. 40 Kilobyte

Risposta corretta.

La risposta corretta è: 48 Kilobyte

Dire che un sistema adotta un *binding dinamico degli indirizzi* significa che:

Scegli un'alternativa:

- a. gli indirizzi relativi usati dalle istruzioni dei programmi in esecuzione possono cambiare da una esecuzione alla successiva.
- b. i programmi che girano su quel sistema usano solo indirizzi relativi, che vengono tradotti in indirizzi assoluti durante l'esecuzione delle istruzioni che usano quegli indirizzi ✔
- c. i programmi che girano su quel sistema usano solo la memoria dinamica per allocare le variabili, ottimizzando così lo spazio occupato da ciascun programma
- d. il sistema operativo traduce gli indirizzi usati dai processi in esecuzione in modo dinamico, ossia solo nel momento in cui tali processi stanno per passare dallo stato "ready to run" allo stato "running"

Risposta corretta.

La risposta corretta è: i programmi che girano su quel sistema usano solo indirizzi relativi, che vengono tradotti in indirizzi assoluti durante l'esecuzione delle istruzioni che usano quegli indirizzi

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 turnaround time medio del sistema relativo ai tre processi risulta lo stesso sia adottando un algoritmo di scheduling FCFS che adottando una algoritmo di scheduling RR?

Scegli un'alternativa:

a. nel caso in cui Burst_P1 = Burst_P2 = Burst_P3 e il quanto di tempo di RR sia inferiore al burst di ciascuno dei tre processi

b. nel caso in cui il burst di ciascuno dei tre processi sia inferiore al quanto di tempo dell'algoritmo RR ✔

c. nel caso in cui Burst_P1 ≤ Burst_P2 ≤ Burst_P3 e il quanto di tempo di RR sia inferiore al burst di tutti e tre processi

d. nel caso in cui il burst di ciascuno dei tre processi sia superiore al quanto di tempo dell'algoritmo RR

Risposta corretta.

La risposta corretta è: nel caso in cui il burst di ciascuno dei tre processi sia inferiore al quanto di tempo dell'algoritmo RR

Di un sistema è noto che la tabella delle pagine più grande del sistema occupa esattamente 4 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 2 Kilobyte.

lo spazio logico del sistema è grande: 1 Megabyte ✔

lo spazio fisico del sistema è grande: 64 Gigabyte ✔

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

Risposta corretta.

La risposta corretta è:

Di un sistema è noto che la tabella delle pagine più grande del sistema occupa esattamente 4 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 2 Kilobyte.

lo spazio logico del sistema è grande: [1 Megabyte]
lo spazio fisico del sistema è grande: [64 Gigabyte]