Iniziato	lunedì, 10 febbraio 2025, 14:00
Stato	Completato
Terminato	lunedì, 10 febbraio 2025, 14:43
Tempo impiegato	42 min. 28 secondi
Valutazione	**17,00** su un massimo di 23,00 (**73,91**%)

Domanda **1**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

In un sistema che implementa la paginazione della memoria, un indirizzo fisico è scritto su 38 bit, l'offset più grande all'interno di una pagina è pari a 7FF, la tabella delle pagine più grande del sistema occupa 8 megabyte.

Quanto è grande lo spazio di indirizzamento logico del sistema?

(selezionate l'opzione di risposta che riporta il ragionamento aritmetico e il risultato corretti)

Scegli un'alternativa:

a. Lo spazio fisico è suddiviso in 2^27 frame, una entry di una tabella delle pagine di questo sistema è grande 2 byte, e lo spazio logico è suddiviso in 2^23/2 = 2^22 entry. Dunque, lo spazio di indirizzamento logico è grande 8 Gigabyte

b. Lo spazio fisico è suddiviso in 2^27 frame, una entry di una tabella delle pagine di questo sistema è grande 4 byte, e lo spazio logico è suddiviso in 2^23/2^2 = 2^21 entry. Dunque, lo spazio di indirizzamento logico è grande 4 Gigabyte

c. Lo spazio fisico è suddiviso in 2^27 frame, una entry di una tabella delle pagine di questo sistema è grande 2 byte, e lo spazio logico è suddiviso in 2^23/2 = 2^22 entry. Dunque, lo spazio di indirizzamento logico è grande 256 Megabyte

d. Lo spazio fisico è suddiviso in 2^27 frame, una entry di una tabella delle pagine di questo sistema è grande 4 byte, e lo spazio logico è suddiviso in 2^23/2^2 = 2^21 entry. Dunque, lo spazio di indirizzamento logico è grande 512 Megabyte ✖

Risposta errata.

La risposta corretta è: Lo spazio fisico è suddiviso in 2^27 frame, una entry di una tabella delle pagine di questo sistema è grande 4 byte, e lo spazio logico è suddiviso in 2^23/2^2 = 2^21 entry. Dunque, lo spazio di indirizzamento logico è grande 4 Gigabyte

Domanda **2**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In un sistema operativo che adotta uno scheduling senza diritto di prelazione, quattro processi arrivano al tempo indicato e consumano la quantità di CPU indicata nella tabella sottostante

Processo	T. di arrivo	Burst
Pa	0	5
Pb	2	3
Pc	4	2
Pd	6	1

Se si usa l'algoritmo di scheduling non preemptive che fornisce le migliori prestazioni possibili per schedulare i 4 processi in tabella:

il waiting time medio è: 2 ✔

il turnaround medio è: 19/4 ✔

il diagramma di GANTT è: (0) ... Pa ... (5) ... Pc ... (7) ... Pd ... (8) ... Pb ... (11) ✔

l'algoritmo usato per le risposte precedenti può soffrire di starvation?

si ✔

18/4

(0) ... Pa ... (5) ... Pd ... (6) ... Pc ... (8) ... Pb ... (11)

no

3

Risposta corretta.

La risposta corretta è:

In un sistema operativo che adotta uno scheduling senza diritto di prelazione, quattro processi arrivano al tempo indicato e consumano la quantità di CPU indicata nella tabella sottostante

Processo	T. di arrivo	Burst
Pa	0	5
Pb	2	3
Pc	4	2
Pd	6	1

Se si usa l'algoritmo di scheduling non preemptive che fornisce le migliori prestazioni possibili per schedulare i 4 processi in tabella:

il waiting time medio è: [2]

il turnaround medio è: [19/4]

il diagramma di GANTT è: [(0) ... Pa ... (5) ... Pc ... (7) ... Pd ... (8) ... Pb ... (11)]

l'algoritmo usato per le risposte precedenti può soffrire di starvation? [si]

Domanda **3**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

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

1: cd /tmp
2: mkdir newfolder
3: echo “ciao” > pippo // crea un nuovo file di nome *pippo* contenente la stringa *ciao*
4: cd newfolder
5: ln -s ../pippo paperino
6: ln -s /tmp/newfolder folder2
7: cp ../pippo 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 *topolino* è: 1 ✔
   2. il link counter di *newfolder* è: 3
   3. l'output del comando 10 è: "ciao"
   4. il link counter di tmp è: aumentato di 1

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

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

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

Risposta corretta.

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

Domanda **4**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In un sistema time-sharing che non usa un algoritmo di sostituzione delle pagine, un processo in esecuzione può passare ad uno stato di wait. Ad esempio perché:

Scegli un'alternativa:

a. perché ha eseguito una wait su un semaforo con valore minore di zero oppure perché ha completato una operazione di I/O

b. perché ha completato una operazione di I/O oppure perché ha indirizzato una pagina non presente in RAM

c. perché ha richiesto una operazione di I/O oppure perché ha indirizzato una pagina non presente in RAM

d. perché ha eseguito una wait su un semaforo con valore minore di zero oppure perché ha richiesto una operazione di I/O ✔

Risposta corretta.

La risposta corretta è: perché ha eseguito una wait su un semaforo con valore minore di zero oppure perché ha richiesto una operazione di I/O

Domanda **5**

Risposta non data

Punteggio max.: 2,00

Si consideri la seguente affermazione: *In un qualsiasi sistema operativo che implementi la memoria virtuale, l'effettivo tempo di esecuzione di un programma dipende molto dal numero di page fault generati dal programma durante la sua esecuzione.* Dite se è vera o falsa, e spiegate perché.

Scegli un'alternativa:

a. l'affermazione è vera: poiché il tempo di gestione di un page fault è comunque alto, un programma che generi molti page fault vede aumentare di molto il suo tempo di esecuzione.

b. l'affermazione è vera: più page fault significa che in media meno pagine del processo occupano spazio in memoria primaria, e il processo può essere fatto girare più velocemente.

c. l'affermazione è falsa: il tempo di esecuzione di un programma dipende principalmente da quello che fa. Le pagine che producono page fault avrebbero dovuto essere comunque caricate in RAM anche se la memoria virtuale non fosse implementata.

d. l'affermazione è falsa: non è tanto il numero di page fault a influenzare in tempo di esecuzione del programma, ma il fatto che le pagine mancanti contengano codice o dati. Se contengono dati infatti vanno prima salvate nell'area di swap, rallentando l'esecuzione del programma.

Risposta errata.

La risposta corretta è: l'affermazione è vera: poiché il tempo di gestione di un page fault è comunque alto, un programma che generi molti page fault vede aumentare di molto il suo tempo di esecuzione.

Domanda **6**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

del codice statico e del codice staticamente rilocabile possiamo dire che:

Scegli un'alternativa:

a. nei sistemi operativi moderni non vengono usati perché per girare in modo efficiente richiederebbero dell'hardware specifico che ne limiterebbe la portabilità

b. nei sistemi operativi moderni non vengono usati perché renderebbero impossibile l'implementazione della memoria virtuale

c. nei sistemi operativi moderni non vengono usati perché producono codice molto meno efficiente di quello dinamicamente rilocabile ✖

d. nei sistemi operativi moderni non vengono usati perché non permetterebbero una implementazione efficiente della paginazione della memoria

Risposta errata.

La risposta corretta è: nei sistemi operativi moderni non vengono usati perché non permetterebbero una implementazione efficiente della paginazione della memoria

Domanda **7**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In un sistema time sharing che usa un algoritmo di sostituzione delle pagine, tra le ragioni per cui si può verificare un context switch tra processi utente troviamo:

Scegli un'alternativa:

a.
1. è stata richiesta una operazione di I/O da parte del processo in CPU
2. il processo in esecuzione si addormenta su un semaforo
3. il processo in esecuzione ha esaurito il suo quanto di tempo
4. un processo con priorità minore di quello in esecuzione entra in RQ

b. ✔
1. è stata richiesta una operazione di I/O da parte del processo in CPU
2. il processo in esecuzione si addormenta su un semaforo
3. il processo in esecuzione ha esaurito il suo quanto di tempo
4. un processo con priorità maggiore di quello in esecuzione entra in RQ

c.
1. è stata completata una operazione di I/O da parte del processo in CPU
2. il processo in esecuzione si addormenta su un semaforo
3. il processo in esecuzione ha esaurito il suo quanto di tempo
4. un processo con priorità maggiore di quello in esecuzione entra in RQ

d.
1. è stata richiesta una operazione di I/O da parte del processo in CPU
2. il processo in esecuzione si sveglia su un semaforo
3. il processo in esecuzione ha esaurito il suo quanto di tempo
4. un processo con priorità maggiore di quello in esecuzione entra in RQ

Risposta corretta.

La risposta corretta è:
1. è stata richiesta una operazione di I/O da parte del processo in CPU
2. il processo in esecuzione si addormenta su un semaforo
3. il processo in esecuzione ha esaurito il suo quanto di tempo
4. un processo con priorità maggiore di quello in esecuzione entra in RQ

Domanda **8**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

il problema della sezione critica può essere risolto in molti modi diversi, ma alcune soluzioni sono migliori di altre. Una qualità auspicabile per una corretta soluzione al problema della sezione critica è:

Scegli un'alternativa:

- a. che non sia basata sul busy waiting, in modo che un processo che cerca di entrare in una sezione critica occupata da un altro processo non sprechi tutto il suo quanto di tempo ✔
- b. che due processi possano stare contemporaneamente in sezione critica se devono solo leggere e non modificare le variabili condivise della sezione critica
- c. che garantisca che il processo in sezione critica esca dalla sezione critica in un tempo finito, in modo che gli altri processi non soffrano di starvation
- d. che il processo che sta uscendo dalla sezione critica possa decidere quale processo può prendere il suo posto, in modo da evitare situazioni di deadlock

Risposta corretta.

La risposta corretta è: che non sia basata sul busy waiting, in modo che un processo che cerca di entrare in una sezione critica occupata da un altro processo non sprechi tutto il suo quanto di tempo

Domanda **9**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In quale caso, e perché, l'allocazione concatenata (senza FAT) dello spazio in memoria secondaria è particolarmente **svantaggiosa** rispetto ad altre tecniche di allocazione viste?

Scegli un'alternativa:

- a. Nel caso di file molto grandi, perché l'accesso sequenziale ai vari dati del file risulta molto più lento che nelle altre forme di allocazione dello spazio su disco
- b. Nel caso di file molto grandi, perché se l'hard disk su cui deve essere memorizzato il file è quasi completamente occupato, il SO potrebbe impiegare molto tempo a trovare un insieme di blocchi liberi in cui allocare il file
- c. Nel caso di file molto grandi, perché in ogni blocco di dati una parte del blocco viene sprecata per memorizzare in puntatore al blocco successivo, risultando così in un notevole spreco di memoria
- d. Nel caso di file molto grandi, perché per leggere i dati contenuti nell'ultima parte del file occorre seguire la catena di blocchi sull'hard disk, ossia effettuare molte operazioni di I/O su un dispositivo lento. ✔

Risposta corretta.

La risposta corretta è: Nel caso di file molto grandi, perché per leggere i dati contenuti nell'ultima parte del file occorre seguire la catena di blocchi sull'hard disk, ossia effettuare molte operazioni di I/O su un dispositivo lento.

Domanda **10**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Un hard disk ha la dimensione di 512 Gigabyte, è formattato in blocchi da 0x1000 byte e adotta una allocazione concatenata (senza FAT) dello spazio su disco. Sull'hard disk è memorizzato un file A della dimensione di 0x4000 byte. Quanti byte del file sono memorizzati nell'ultimo blocco di dati del file? (selezionate l'opzione di risposta che riporta il calcolo corretto del valore cercato)

Scegli un'alternativa:

a. 2^39/2^12 = 2^27 = = > 4 byte per scrivere un puntatore a blocco. Dunque, l'ultimo blocco di A conterrà 2^14/4 = 4096 byte di A

b. 2^39/2^15 = 2^24 = = > 3 byte per scrivere un puntatore a blocco. Dunque, l'ultimo blocco di A conterrà 3 * 2^15/2^13 = 12 byte di A

c. 2^39/2^13 = 2^26 = = > 4 byte per scrivere un puntatore a blocco. Dunque, l'ultimo blocco di A conterrà 2^15/4 = 8192 byte di A

d. 2^39/2^12 = 2^27 = = > 4 byte per scrivere un puntatore a blocco. Dunque, l'ultimo blocco di A conterrà 4 * 2^14/2^12 = 16 byte di A ✔

Risposta corretta.

La risposta corretta è: 2^39/2^12 = 2^27 = = > 4 byte per scrivere un puntatore a blocco. Dunque, l'ultimo blocco di A conterrà 4 * 2^14/2^12 = 16 byte di A

Domanda **11**

Risposta corretta

Punteggio ottenuto 3,00 su 3,00

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

Risposta corretta.

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
```