Iniziato	giovedì, 12 febbraio 2026, 14:07
Stato	Completato
Terminato	giovedì, 12 febbraio 2026, 14:51
Tempo impiegato	44 min. 20 secondi
Valutazione	**15,00** su un massimo di 23,00 (**65,22%**)

Domanda **1**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

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. 40 Kilobyte

b. 24 Kilobyte

c. 48 Kilobyte

d. 32 Kilobyte ✖

Risposta errata.

La risposta corretta è: 48 Kilobyte

Domanda **2**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Che vantaggio da l'uso dei thread al posto dei processi?

Scegli un'alternativa:

a. un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, i vari thread non hanno bisogno di ricorrere alla memoria virtuale, la creazione di un nuovo peer thread richiede molto meno tempo della corrispondente operazione sui processi.

b. un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, il context switch tra peer thread e la creazione di un nuovo peer thread richiedono molto meno tempo delle corrispondenti operazioni sui processi. ✔

c. un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, i vari thread non hanno bisogno di ricorrere alla memoria virtuale, e l'esecuzione dei vari thread richiede meno tempo di un insieme di processi che esegua lo stesso codice dei thread.

d. un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, i vari thread non devono usare meccanismi di sincronizzazione per accedere alle variabili condivise, e il context switch tra peer thread richiede molto meno tempo della corrispondente operazione tra processi.

Risposta corretta.

La risposta corretta è: un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, il context switch tra peer thread e la creazione di un nuovo peer thread richiedono molto meno tempo delle corrispondenti operazioni sui processi.

Domanda **3**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Se si verifica un page fault:

Scegli un'alternativa:

a. significa che il processo running ha indirizzato una pagina non appartenente al suo spazio di indirizzamento. Il processo viene tolto dall'esecuzione e il sistema operativo incomincia a recuperare la pagina che era stata indirizzata. A operazione completata il processo viene riportato in coda di ready

b. significa che il processo running ha indirizzato una pagina non presente in RAM. Il processo viene tolto dall'esecuzione e il sistema operativo incomincia a recuperare la pagina che era stata indirizzata. A operazione completata il processo viene riportato in esecuzione

c. significa che un processo in coda di ready ha indirizzato una pagina non presente in RAM. Il processo viene tolto dalla coda e il sistema operativo incomincia a recuperare la pagina che era stata indirizzata. A operazione completata il processo viene riportato in coda di ready

d. significa che il processo running ha indirizzato una pagina non presente in RAM. Il processo viene tolto dall'esecuzione e il sistema operativo incomincia a recuperare la pagina che era stata indirizzata. A operazione completata il processo viene riportato in coda di ready ✔

Risposta corretta.

La risposta corretta è: significa che il processo running ha indirizzato una pagina non presente in RAM. Il processo viene tolto dall'esecuzione e il sistema operativo incomincia a recuperare la pagina che era stata indirizzata. A operazione completata il processo viene riportato in coda di ready

Domanda **4**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

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 8 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. 256 Megabyte ✖
- b. 64 Megabyte
- c. 128 Megabyte
- d. 512 Megabyte

Risposta errata.

La risposta corretta è: 128 Megabyte

Domanda **5**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

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

1: cd /tmp
2: mkdir newfolder
3: 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: 2
2. il link counter di *newfolder* è: 3
3. l'output del comando 10 è: no such file or directory
4. il link counter di tmp è: aumentato di 1

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: 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

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 2

Risposta errata.

La risposta corretta è:

1. il link-counter dell'i-node di *paperino* 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 **6**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Assumendo che siano presenti in un sistema almeno due processi utente, quattro possibili ragioni per cui in un moderno sistema operativo time sharing che implementa la memoria virtuale si può verificare un context switch tra due processi utente sono:

Scegli un'alternativa:

a.
1. il processo running genera una trap e viene terminato
2. il processo running inizia una operazione di I/O
3. entra in coda di wait un processo con priorità maggiore di quello in stato running
4. il processo running esegue una wait e si addormenta sul semaforo

b.
1. il processo running genera una trap e viene terminato
2. il processo running inizia una operazione di I/O
3. entra in coda di ready un processo con priorità maggiore di quello in stato running
4. il processo running esegue una wait su un semaforo con valore > 0

c. ✔
1. il processo running genera una trap e viene terminato
2. il processo running inizia una operazione di I/O
3. entra in coda di ready un processo con priorità maggiore di quello in stato running
4. il processo running esegue una wait e si addormenta sul semaforo

d.
1. il processo running genera una trap e viene terminato
2. il processo running ha appena completato una operazione di I/O
3. entra in coda di ready un processo con priorità maggiore di quello in stato running
4. il processo running esegue una wait e si addormenta sul semaforo

Risposta corretta.

La risposta corretta è:
1. il processo running genera una trap e viene terminato
2. il processo running inizia una operazione di I/O
3. entra in coda di ready un processo con priorità maggiore di quello in stato running
4. il processo running esegue una wait e si addormenta sul semaforo

Domanda **7**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Ricostruite il codice della system call Signal:

```
signal(semaforo *S) {
    S->valore++;                          ✔
    if S->valore <= 0                     ✔
     {
      togli un processo P da S->waiting_list;   ✔
      wakeup(P);                          ✔
     }
}
```

if S->valore < 0

sleep();

S->valore--;

aggiungi questo processo a S->waiting_list;

if S->valore == 0

Risposta corretta.

La risposta corretta è:
Ricostruite il codice della system call Signal:

```
signal(semaforo *S) {
  [S->valore++;]
  [if S->valore <= 0]
    {
    [togli un processo P da S->waiting_list;]
    [wakeup(P);]
    }
}
```

Domanda **8**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In quale/i caso/i un processo in coda di ready può decidere di passare allo stato running?

Scegli un'alternativa:

a. in tutti gli algoritmi di scheduling non premeptive, dove un processo può decidere autonomamente quanto tempo passare in coda di ready

b. mai, è il Sistema Operativo che sposta i processi da uno stato all'altro. ✔

c. quando un processo esce da un qualsiasi stato di waiting

d. quando un processo si trova in testa alla coda di ready

Risposta corretta.

La risposta corretta è: mai, è il Sistema Operativo che sposta i processi da uno stato all'altro.

Domanda **9**

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 = 40, b = 50, c = 60, d = 70;
n = fork();
      if ( n == 0)
         {a = 45; b = 55;
          pid1 = getppid();
          printf("%d", pid1);
          exit(0);}
      else
         {c = 65; d = 75;
          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 [45] ✔

il valore della variabile c vista dal processo figlio subito prima della sua exit è [60] ✔

il valore della variabile b vista dal processo padre subito prima della sua exit è: [50] ✔

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

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

del risultato della wait possiamo dire che: pid1 [<] ✔ pid3

[<] [55] [65] [45] [=] [75] [>] [50] [70] [40] [60]

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 = 40, b = 50, c = 60, d = 70;
n = fork();
      if ( n == 0)
         {a = 45; b = 55;
          pid1 = getppid();
          printf("%d", pid1);
          exit(0);}
      else
         {c = 65; d = 75;
          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 [45]
il valore della variabile c vista dal processo figlio subito prima della sua exit è [60]
il valore della variabile b vista dal processo padre subito prima della sua exit è: [50]
il valore della variabile d vista dal processo padre subito prima della sua exit è: [75]
all'esecuzione delle due printf vale la seguente relazione: pid1 [=] pid2
del risultato della wait possiamo dire che: pid1 [<] pid3
```

Domanda **10**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

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: 64 Gigabyte ✖

lo spazio fisico del sistema è grande: 4 Megabyte ✖

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

Risposta errata.

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]

Domanda **11**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Gli **svantaggi** delle librerie statiche sono:

Scegli un'alternativa:

a.
1. Non possono essere condivise tra più processi, per cui occupano più spazio in RAM di quelle dinamiche
2. Vengono caricate in RAM anche se non sono chiamate, per cui occupano comunque spazio in RAM e i processi partono più lentamente
3. Non possono essere aggiornate

b. ✔
1. Non possono essere condivise tra più processi, per cui occupano più spazio in RAM di quelle dinamiche
2. Vengono caricate in RAM anche se non sono chiamate, per cui occupano comunque spazio in RAM e i processi partono più lentamente
3. Se vengono aggiornate occorre ricompilare i programmi che le usano

c.
1. Per poter essere condivise tra più processi è necessario l'uso di un segmento di shared memory
2. Vengono caricate in RAM anche se non sono chiamate, per cui occupano comunque spazio in RAM e i processi partono più lentamente
3. Se vengono aggiornate occorre ricompilare i programmi che le usano

d.
1. Non possono essere condivise tra più processi, per cui occupano più spazio in RAM di quelle dinamiche
2. Vengono caricate in RAM anche se non sono chiamate, per cui non possono essere usate con la memoria virtuale
3. Se vengono aggiornate occorre ricompilare i programmi che le usano

Risposta corretta.

La risposta corretta è:
1. Non possono essere condivise tra più processi, per cui occupano più spazio in RAM di quelle dinamiche
2. Vengono caricate in RAM anche se non sono chiamate, per cui occupano comunque spazio in RAM e i processi partono più lentamente
3. Se vengono aggiornate occorre ricompilare i programmi che le usano