Iniziato	martedì, 27 gennaio 2026, 09:05
Stato	Completato
Terminato	martedì, 27 gennaio 2026, 09:51
Tempo impiegato	45 min. 43 secondi
Valutazione	**24,00** su un massimo di 30,00 (**80**%)

Domanda **1**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Un sistema ha un tempo di accesso in RAM di 120 ns, adotta un TLB con un tempo di accesso di 10 ns e un hit rate del 95%, usa una paginazione a due livelli e non ha bisogno di usare un algoritmo di rimpiazzamento delle pagine. Qual è il tempo medio di accesso alla RAM (*medium access time - mat*) del sistema? (per semplicità in caso di miss si ignori il costo di interrogazione del TLB)

Scegli un'alternativa:

- a. mat = 141,5 ns ✔
- b. mat = 147,5 ns
- c. mat = 130,5 ns
- d. mat = 135,5 ns

Risposta corretta.

La risposta corretta è: mat = 141,5 ns

Domanda **2**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In un sistema che adotta la paginazione della memoria, un indirizzo logico è scritto su "P" bit, e il numero di pagina è scritto su "k" bit. Lo spazio fisico è invece suddiviso in 2^24 frame.

Possiamo quindi dire che:

un frame del sistema è grande: 2^(P-k) ✔ byte

lo spazio fisico del sistema è grande: 2^(24+P-k) ✔ byte

la tabella delle pagine più grande del sistema ha una dimensione di: (2^k) * 3 ✔ byte

il sistema dovrà implementare la memoria virtuale solo se: P > (24+P-k) ✔

2^(P+24) (2^24) * 3 2^(24-k) 2^(24-P+k) P < (24+P-k)

Risposta corretta.

La risposta corretta è:
In un sistema che adotta la paginazione della memoria, un indirizzo logico è scritto su "P" bit, e il numero di pagina è scritto su "k" bit. Lo spazio fisico è invece suddiviso in 2^24 frame.

Possiamo quindi dire che:

un frame del sistema è grande: [2^(P-k)] byte

lo spazio fisico del sistema è grande: [2^(24+P-k)] byte

la tabella delle pagine più grande del sistema ha una dimensione di: [(2^k) * 3] byte

il sistema dovrà implementare la memoria virtuale solo se: [P > (24+P-k)]

Domanda **3**

Risposta non data

Punteggio max.: 2,00

Un algoritmo di scheduling a code multiple con retroazione:

Scegli un'alternativa:

a. usa più code di ready, gestite ciascuna con una diversa politica di scheduling. Un processo può essere spostato da una coda all'altra in base a come si è comportato l'ultima volta che gli è stata assegnata la CPU

b. usa più code di ready, in cui i processi vengono inseriti a seconda che siano processi CPU o I/O bound, e processi in foreground o in background

c. usa più code di ready, gestite ciascuna con una diversa politica di scheduling. Un processo può essere retrocesso a una coda con priorità inferiore se non ha consumato completamente il suo ultimo quanto di tempo

d. usa più code di ready, gestite ciascuna con una diversa politica di scheduling. Un processo può essere promosso a una coda con priorità superiore se ha consumato completamente il suo ultimo quanto di tempo

Risposta errata.

La risposta corretta è: usa più code di ready, gestite ciascuna con una diversa politica di scheduling. Un processo può essere spostato da una coda all'altra in base a come si è comportato l'ultima volta che gli è stata assegnata la CPU

Domanda **4**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Ricostruite il codice della system call Wait:

```
wait(semaforo *S) {
    S->valore--;
    if S->valore < 0
    {
      aggiungi questo processo a S->waiting_list;
      sleep();
    }
}
```

wakeup(P);

if S->valore <= 0

if S->valore == 0

S->valore++;

togli un processo P da S->waiting_list;

Risposta corretta.

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

```
wait(semaforo *S) {
  [S->valore--;]
  [if S->valore < 0]
    {
    [aggiungi questo processo a S->waiting_list;]
    [sleep();]
    }
  }
```

Domanda **5**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Dal punto di vista della memoria virtuale, il modo in cui i programmi accedono ai dati in RAM può pesantemente influire sulla velocità di esecuzione dei programmi stessi. Di questa affermazione si può dire che:

Scegli un'alternativa:

a. è vera, in quanto l'accesso ai dati nell'area di swap richiede molto tempo, dato che l'area di swap è implementata su un dispositivo di memorizzazione lento

b. è falsa, in quanto il tempo medio di esecuzione di un programma dipende prevalentemente dalla quantità di tempo necessario a elaborare i dati, e non dall'ordine con cui questi dati sono acceduti

c. è vera, in quanto dal modo con cui i dati vengono indirizzati dipende il numero di page fault generati, e quindi il tempo medio di esecuzione dei programmi ✔

d. è falsa, in quanto il tempo medio di esecuzione di un programma dipende dal numero di page fault che genera, e non dal modo con cui accede ai suoi dati

Risposta corretta.

La risposta corretta è: è vera, in quanto dal modo con cui i dati vengono indirizzati dipende il numero di page fault generati, e quindi il tempo medio di esecuzione dei programmi

Domanda **6**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Un hard disk ha la dimensione di 512 Gigabyte ed è formattato in blocchi da 0x1000 byte. Qual è la dimensione della FAT dell'hard disk? (selezionate l'opzione di risposta che riporta l'espressione aritmetica corretta "num. entry * dim. entry" per il calcolo della dimensione della FAT)

Scegli un'alternativa:

a. $2^{26} * 2^{3}$ = 512 Megabyte

b. $2^{27} * 2^{2}$ = 512 Megabyte ✔

c. $2^{26} * 2^{2}$ = 256 Megabyte

d. $2^{27} * 2$ = 256 Megabyte

Risposta corretta.

La risposta corretta è: $2^{27} * 2^{2}$ = 512 Megabyte

Domanda **7**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

Nello Unix, quale/quali dei seguenti comandi modifica il valore del *link counter* dell'index-node associato al file di testo X? (si assuma che tutti i comandi vengono eseguiti correttamente)

1) ln X Y

2) ln -s X Y

3) rm X Y

4) ls X Y

Scegli un'alternativa:

- a. i comandi 2) e 3)
- b. i comandi 1) e 3) ✔
- c. i comandi 1) e 4)
- d. i comandi 2) e 4)

Risposta corretta.

La risposta corretta è: i comandi 1) e 3)

Domanda **8**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

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 = 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

85 | 90 | 65 | 60 | = | 95 | 75 | < | 80 | 70 | >

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 = 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
```

Domanda **9**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

Del binding dinamicamente rilocabile possiamo dire che (scegliete l'unica opzione completamente corretta):

Scegli un'alternativa:

a. è il binding in cui gli indirizzi nascono relativi e rimangono relativi durante l'intera esecuzione del programma, e un suo **vantaggio** fondamentale è che la ricompilazione dei programmi dopo eventuali modifiche risulta particolarmente efficiente

b. è il binding in cui gli indirizzi nascono assoluti e rimangono assoluti durante l'intera esecuzione del programma, e un suo **vantaggio** fondamentale è che lo spostamento da un punto all'altro della RAM durante l'esecuzione ha un costo molto limitato

c. è il binding in cui gli indirizzi nascono relativi e rimangono relativi durante l'intera esecuzione del programma, e un suo **svantaggio** fondamentale è che richiede dell'hardware specifico affinché la traduzione degli indirizzi da logici a fisici non costituisca un overhead eccessivo

d. è il binding in cui gli indirizzi nascono relativi e vengono trasformati in assoluti durante il caricamento in RAM del programma, e un suo **svantaggio** fondamentale è che se si vuole spostare il programma da un punto all'altro della RAM occorre effettuarne di nuovo il caricamento. ✖

Risposta errata.

La risposta corretta è: è il binding in cui gli indirizzi nascono relativi e rimangono relativi durante l'intera esecuzione del programma, e un suo **svantaggio** fondamentale è che richiede dell'hardware specifico affinché la traduzione degli indirizzi da logici a fisici non costituisca un overhead eccessivo

Domanda **10**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In un sistema operativo che adotta uno scheduling con 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	8
Pb	2	1
Pc	4	2
Pd	6	3

Qual è il waiting time medio ottenuto per lo scheduling dei quattro processi della tabella se si usa l'algoritmo di scheduling preemptive che fornisce il miglior turnaround time possibile? Qual è il corrispondente diagramma di GANTT?

Scegli un'alternativa:

a. Diagramma di GANTT: (0) ... Pa ... (2) ... Pb ... (3) ...Pc... (4) ... Pa ... (6) ... Pd ... (9) ... Pa ... (14)
Waiting time medio = 7/4

b. Diagramma di GANTT: (0) ... Pa ... (2) ... Pb ... (3) ... Pd ... (6) ... Pc ... (8) ... Pa ...(14)
Waiting time medio = 7/4

c. Diagramma di GANTT: (0) ... Pa ... (2) ... Pb ... (3) ... Pa ... (4) ... Pc ... (6) ... Pd ... (9) ... Pa ... (14) ✔
Waiting time medio = 6/4

d. Diagramma di GANTT: (0) ... Pa ... (2) ... Pb ... (3) ... Pc ... (5) ... Pd ... (8) ... Pa ...(14)
Waiting time medio = 6/4

Risposta corretta.

La risposta corretta è:
Diagramma di GANTT: (0) ... Pa ... (2) ... Pb ... (3) ... Pa ... (4) ... Pc ... (6) ... Pd ... (9) ... Pa ... (14)

Waiting time medio = 6/4

Domanda **11**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

In un sistema paginato è noto che lo spreco di memoria primaria dovuto alla frammentazione interna è in media di circa 1 Kbyte per processo, e un indirizzo logico è scritto su 28 bit. Se la tabella delle pagine più grande di questo sistema è grande 256 Kilobyte, quanto può essere grande al massimo lo spazio di indirizzamento fisico del sistema?

Scegli un'alternativa:

- a. 64 Megabyte
- b. 512 Megabyte
- c. 256 Megabyte
- d. 128 Megabyte ✔

Risposta corretta.

La risposta corretta è: 128 Megabyte

Domanda **12**

Risposta errata

Punteggio ottenuto 0,00 su 2,00

Un hard disk ha la dimensione di 512 Gigabyte, è formattato in blocchi da 0x200 byte e adotta una qualche forma di allocazione indicizzata dello spazio su disco. Sull'hard disk è memorizzato un file A della dimensione di 150 Kbyte. Quante operazioni di I/O sono necessarie per leggere l'ultimo blocco di dati del file, assumendo già in RAM tutti gli attributi del file? (selezionate l'opzione di risposta che riporta il ragionamento numerico corretto)

Scegli un'alternativa:

a. 2^39/2^10 = 2^29, dunque un blocco indice di questo hard disk può tenere traccia di un massimo di 128 Kbyte di dati. Assumendo una allocazione indicizzata a più livelli, ci vogliono 3 livelli di indirezione per registrare l'ultimo blocco di dati del file, e dunque, se è gia in RAM il numero blocco indice più esterno, saranno necessarie in tutto 4 operazioni di lettura su disco

b. 2^39/2^9 = 2^30, dunque un blocco indice di questo hard disk può tenere traccia di un massimo di 64 Kbyte di dati. Assumendo una allocazione indicizzata a schema concatenato, ci vogliono 3 blocchi indice per registrare l'ultimo blocco di dati del file, e dunque, se è gia in RAM il numero del primo blocco indice, saranno necessarie in tutto 4 operazioni di lettura su disco

c. 2^39/2^10 = 2^29, dunque un blocco indice di questo hard disk può tenere traccia di un massimo di 128 Kbyte di dati. Assumendo una allocazione indicizzata a schema concatenato, ci vogliono 2 blocchi indice per registrare l'ultimo blocco di dati del file, e dunque, se è gia in RAM il numero del primo blocco indice, saranno necessarie in tutto 3 operazioni di lettura su disco

d. 2^39/2^9 = 2^30, dunque un blocco indice di questo hard disk può tenere traccia di un massimo di 64 Kbyte di dati. Assumendo una allocazione indicizzata a più livelli, ci vogliono 3 livelli di indirezione per registrare l'ultimo blocco di dati del file, e dunque, se è gia in RAM il numero del blocco indice più esterno, saranno necessarie in tutto 4 operazioni di lettura su disco ✖

Risposta errata.

La risposta corretta è: 2^39/2^9 = 2^30, dunque un blocco indice di questo hard disk può tenere traccia di un massimo di 64 Kbyte di dati. Assumendo una allocazione indicizzata a schema concatenato, ci vogliono 3 blocchi indice per registrare l'ultimo blocco di dati del file, e dunque, se è gia in RAM il numero del primo blocco indice, saranno necessarie in tutto 4 operazioni di lettura su disco

Domanda **13**

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: 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 2

b. 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

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: 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

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

Domanda **14**

Risposta corretta

Punteggio ottenuto 2,00 su 2,00

L'allocazione indicizzata dello spazio in memoria secondaria è particolarmente **svantaggiosa** quando:

Scegli un'alternativa:

a. in nessun caso, infatti l'allocazione indicizzata, nelle sue varianti commerciali, è proprio quella impiegata sia nei sistemi Unix che nei sistemi Windows

b. nel caso di file molto grandi, perché la perdita del blocco indice produce la perdita dell'intero file

c. nel caso di file che devono essere acceduti in modo sequenziale, in quanto l'allocazione indicizzata è progettata esplicitamente per favorire l'accesso diretto ai dati del file

d. quando si deve allocare spazio per file grandi meno di un blocco di dati, perché un intero blocco indice viene usato per tenere traccia di quell'unico blocco, e quindi quasi tutto il blocco indice rimane inutilizzato ✔

Risposta corretta.

La risposta corretta è: quando si deve allocare spazio per file grandi meno di un blocco di dati, perché un intero blocco indice viene usato per tenere traccia di quell'unico blocco, e quindi quasi tutto il blocco indice rimane inutilizzato

Domanda **15**

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. scade il quanto di tempo del processo running
2. entra in coda di ready un processo con priorità maggiore di quello in stato running
3. il processo running ha appena completato una operazione di I/O
4. il processo running esegue una wait e si addormenta sul semaforo

b.
1. scade il quanto di tempo del processo running
2. entra in coda di ready un processo con priorità maggiore di quello in stato running
3. il processo running inizia una operazione di I/O
4. il processo running esegue una signal e si addormenta sul semaforo

c.
1. scade il quanto di tempo del processo running
2. entra in coda di ready un processo con priorità inferiore di quello in stato running
3. il processo running inizia una operazione di I/O
4. il processo running esegue una wait e si addormenta sul semaforo

d. ✔
1. scade il quanto di tempo del processo running
2. entra in coda di ready un processo con priorità maggiore di quello in stato running
3. il processo running inizia una operazione di I/O
4. il processo running esegue una wait e si addormenta sul semaforo

Risposta corretta.

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