Iniziato	martedì, 8 febbraio 2022, 10:00
Stato	Completato
Terminato	martedì, 8 febbraio 2022, 10:40
Tempo impiegato	39 min. 55 secondi
Valutazione	**7,5** su un massimo di 20,0 (**38**%)

Domanda 1

Risposta errata

Punteggio ottenuto 0,0 su 2,0

Selezionate l'opzione che descrive la corretta implementazione della signal e il significato della system call usata per implementarla.

Scegli un'alternativa:

a.

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

La wakeup(Q) rimette Q nella Ready Queue

---

b.

```
signal(semaforo *S) {
  S->valore++;
  if S->valore <= 0 {
        metti un processo Q in S->waiting_list;
        wakeup(P);}
}
```

La wakeup(P) rimette P nella Ready Queue

---

c.

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

La wakeup(Q) manda Q in esecuzione

---

d.

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

La wakeup(Q) rimette Q nella Ready Queue

Risposta errata.

La risposta corretta è:

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

La wakeup(Q) rimette Q nella Ready Queue

Domanda 2

Risposta corretta

Punteggio ottenuto 2,0 su 2,0

Dell'allocazione concatenata (senza FAT) possiamo dire che tra i suoi **vantaggi** e i suoi **svantaggi** troviamo:

Scegli un'alternativa:

a.
1. l'uso di cluster di blocchi aumenta la quantità di memoria che si spreca per memorizzare i puntatori che implementano la catena di blocchi, ma diminuisce la frammentazione interna
2. non ha bisogno di usare blocchi adiacenti
3. se si rompe la catena di blocchi è difficile o impossibile recuperare tutti i dati di un file

b.
1. rispetto all'allocazione indicizzata spreca poco spazio per tenere traccia di tutti i blocchi di dati di un file, ed è particolarmente adatta per file piccoli,
2. non ha bisogno di usare blocchi adiacenti,
3. è molto inefficiente per l'accesso diretto ai dati dei file molto grandi

✓

c.
1. spreca poco spazio per tenere traccia di tutti i blocchi di dati di un file, ed è particolarmente adatta per l'accesso diretto a file molto grandi,
2. non ha bisogno di usare blocchi adiacenti,
3. se si rompe la catena di blocchi è difficile o impossibile recuperare tutti i dati di un file

d.
1. è particolarmente adatta per memorizzare file di piccole dimensioni
2. non gestisce bene il caso in cui un file cresce di dimensioni e bisogna trovare nuovi blocchi per allocarlo
3. è molto inefficiente per l'accesso diretto ai dati dei file molto grandi

Risposta corretta.

La risposta corretta è:
1. rispetto all'allocazione indicizzata spreca poco spazio per tenere traccia di tutti i blocchi di dati di un file, ed è particolarmente adatta per file piccoli,
2. non ha bisogno di usare blocchi adiacenti,
3. è molto inefficiente per l'accesso diretto ai dati dei file molto grandi

Domanda 3

Risposta errata

Punteggio ottenuto -1,5 su 3,0

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: ln -s 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 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

- b.
  1. il link-counter dell'i-node di *paperino* è: 2
  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

- c.
  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 nuovo collegamento alla cartella newfolder

✗

- d.
  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" seguito da "salve"
  4. il comando 6 da come risultato: un errore perché non sono ammessi hard link tra cartelle

Risposta errata.

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" seguito da "salve"
4. il comando 6 da come risultato: un errore perché non sono ammessi hard link tra cartelle

Domanda 4

Risposta corretta

Punteggio ottenuto 3,0 su 3,0

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	8
Pc	4	2
Pd	6	1

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 ... (4) ... Pb ... (7) ... Pc ... (10) ... Pa ... (11) ... Pd ... (19)

Waiting time medio = 4

b.
Diagramma di GANTT: (0) ... Pa ... (4) ... Pc ... (6) ... Pd ... (7) ... Pa ... (11) ... Pb ... (19)

Waiting time medio = 3 ✓

c.
Diagramma di GANTT: (0) ... Pa ... (4) ... Pb ... (7) ... Pd ... (9) ... Pa ... (11) ... Pc ... (19)

Waiting time medio = 3

d.
Diagramma di GANTT: (0) ... Pa ... (4) ... Pc ... (6) ... Pa ... (10) ... Pd ... (11) ... Pc ... (19)

Waiting time medio = 4

Risposta corretta.

La risposta corretta è:
Diagramma di GANTT: (0) ... Pa ... (4) ... Pc ... (6) ... Pd ... (7) ... Pa ... (11) ... Pb ... (19)

Waiting time medio = 3

Domanda 5

Risposta corretta

Punteggio ottenuto 2,0 su 2,0

Degli algoritmi di sostituzione delle pagine possiamo dire che (scegliete l'unica opzione completamente corretta):

Scegli un'alternativa:

- a. l'algoritmo di sostituzione ottimale (detto OPT o MIN) è detto così perché sceglie come pagina vittima di un processo quella che non verrà mai più indirizzata dal processo stesso
- b. LRU è uno dei migliori algoritmi di rimpiazzamento, ma non viene implementato perché richiederebbe un supporto hardware normalmente non disponibile ✓
- c. l'algoritmo della seconda chance è una approssimazione di LRU implementata usando il dirty bit per distinguere le pagine modificate da quelle non modificate
- d. l'algoritmo della seconda chance migliorata è chiamato così perché, al contrario dell'algoritmo della seconda chance, non soffre dell'anomalia di Belady

Risposta corretta.

La risposta corretta è: LRU è uno dei migliori algoritmi di rimpiazzamento, ma non viene implementato perché richiederebbe un supporto hardware normalmente non disponibile

Domanda 6

Risposta corretta

Punteggio ottenuto 2,0 su 2,0

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 esegue una wait senza addormentarsi sul semaforo
2. il processo running genera una trap e viene terminato
3. il processo running genera page fault
4. il processo running termina di eseguire il suo codice

b.
1. il processo running esegue una wait e si addormenta sul semaforo
2. il processo running genera una trap e viene terminato
3. il processo running genera page fault
4. il processo running termina di eseguire il suo codice

c.
1. il processo running esegue una wait e si addormenta sul semaforo
2. il processo running genera una trap e viene rimesso in coda di ready
3. il processo running genera page fault
4. il processo running termina di eseguire il suo codice

d.
1. il processo running esegue una wait e si addormenta sul semaforo
2. il processo running genera una trap e viene terminato
3. il processo running genera page fault
4. il processo in coda di ready termina di eseguire il suo codice

Risposta corretta.

La risposta corretta è:
1. il processo running esegue una wait e si addormenta sul semaforo
2. il processo running genera una trap e viene terminato
3. il processo running genera page fault
4. il processo running termina di eseguire il suo codice

Domanda 7

Risposta errata

Punteggio ottenuto 0,0 su 1,0

In quale caso l'accesso in lettura ad un file memorizzato su un sistema RAID non è più veloce che se il file fosse memorizzato su un normale hard disk?

Scegli un'alternativa:

- a. mai: infatti, i sistemi RAID sono stati progettati proprio per fornire più elevate velocità di accesso ai dati che contengono (oltre che per garantire una maggiore affidabilità)
- b. quando il file è memorizzato su uno più blocchi appartenenti a strip contenuti sullo stesso disco del RAID (e il RAID usato non è di tipo 01/10, poiché in questo caso, se il file è memorizzato su almeno due strip, si può sfruttare il disco di mirroring)
- c. quando il file occupa molti strip, perché allora neppure la presenza di dischi di mirroring può mitigare la necessità di accedere più volte al contenuto dei vari dischi per leggere tutti gli strip in cui è suddiviso il file ✗
- d. quando il file è memorizzato su dischi che contengono anche strip di parità, come nel livello 5, perché in questo caso la ricostruzione dei dati di cui è composto il file richiede una certa quantità di lavoro in più da parte del controller del RAID

Risposta errata.

La risposta corretta è: quando il file è memorizzato su uno più blocchi appartenenti a strip contenuti sullo stesso disco del RAID (e il RAID usato non è di tipo 01/10, poiché in questo caso, se il file è memorizzato su almeno due strip, si può sfruttare il disco di mirroring)

Domanda 8

Risposta errata

Punteggio ottenuto 0,0 su 3,0

In un sistema paginato è noto che lo spreco di memoria primaria dovuto alla frammentazione interna è in media di circa 4 Kbyte per processo. Un indirizzo fisico è scritto su 29 bit e lo spazio di indirizzamento logico è 8 volte quello fisico.

Qual è la dimensione della tabella delle pagine più grande di questo sistema? (selezionate l'opzione di risposta che riporta il ragionamento aritmetico e il risultato corretti)

Scegli un'alternativa:

- a. Ogni entry della PT più grande del sistema deve essere grande almeno 2 byte, e dunque la PT sarà grande 2 * 2^16 = 128 Kbyte (circa)
- b. Ogni entry della PT più grande del sistema deve essere grande almeno 2 byte, e dunque la PT sarà grande 2 * 2^19 = 1024 Kbyte (circa)
- c. Ogni entry della PT più grande del sistema deve essere grande almeno 3 byte, e dunque la PT sarà grande 3 * 2^19 = 1536 Kbyte (circa) ✗
- d. Ogni entry della PT più grande del sistema deve essere grande almeno 3 byte, e dunque la PT sarà grande 3 * 2^16 = 192 Kbyte (circa)

Risposta errata.

La risposta corretta è: Ogni entry della PT più grande del sistema deve essere grande almeno 2 byte, e dunque la PT sarà grande 2 * 2^19 = 1024 Kbyte (circa)

Domanda 9

Risposta errata

Punteggio ottenuto 0,0 su 2,0

In quali casi e perché si usa una Inverted Page Table?

Scegli un'alternativa:

- a. Le IPT si usano quando non è disponibile il TLB, e dunque la traduzione degli indirizzi da logici a fisici risulterebbe troppo onerosa usando delle normali page table
- b. Le IPT si usano nei sistemi con spazi di indirizzamento logici molto grandi per evitare la paginazione a più livelli, che rende la traduzione degli indirizzi da logici a fisici molto onerosa quando non è possibile ricorrere al TLB
- c. Le IPT si usano quando abbiamo degli spazi di indirizzamento logico molto piccoli, perché in questo modo possiamo usare un'unica page table per tutti i processi, anziché averne una per ciascun processo ✗
- d. Le IPT si usano nei sistemi con spazi di indirizzamento fisico piccoli, in questo modo viene sprecato meno spazio da parte delle page table dei processi, attraverso la paginazione a più livelli

Risposta errata.

La risposta corretta è: Le IPT si usano nei sistemi con spazi di indirizzamento logici molto grandi per evitare la paginazione a più livelli, che rende la traduzione degli indirizzi da logici a fisici molto onerosa quando non è possibile ricorrere al TLB